Universidade de Aveiro
2009
Departamento de Electrónica, Telecomunicações e Informática

Sandra Catarina Otero da Silva e Costa

Sistema servo-mecanismo didáctico com motores DC e passo-a-passo.

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Dr. Telmo Cunha, Professor auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Dr. António Pereira de Melo, Professor catedrático do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri

presidente

Professor Doutor Alexandre Manuel Moutela Nunes Mota
Professor Associado, Universidade de Aveiro

Professor Doutor Pedro António Amado Assunção
Professor Coordenador, Esc. Sup. de Tec. e Gestão de Leiria do Inst. Politécnico de Leiria

Professor Doutor Telmo Reis Cunha
Professor Auxiliar, Universidade de Aveiro

Professor Doutor António Ferreira Pereira de Melo
Professor Catedrático, Universidade de Aveiro

agradecimentos Aos meus orientadores pelo tempo e esclarecimentos prestados. E a todos os que me ajudaram de forma directa ou indirecta a concluir este projecto.

palavras-chave Controlo, motor DC, motor passo-a-passo, hardware, software.

resumo O presente trabalho propõe-se a criar uma plataforma de controlo de motores DC e passo a passo, de forma a contribuir para um laboratório no DET.
Para tal foi desenvolvido uma plataforma onde fosse possível de forma simples alterar o acoplamento dos motores, de forma a fazer vários estudos sobre os mesmos.

keywords Control, DC motor, stepper motor, hardware, software.

abstract The present work proposes to create one platform to control a DC motor and a stepper motor, to contribute for a laboratory in the DET. This platform was developed so that it is possible in a simple way to modify the coupling of the motors to make some other studies on the same ones.

Índice

Índice de figuras

Título da Dissertação:

Sistema servo-mecanismo didáctico com motores DC e passo-a-passo.

Índice de Tabelas

Índice de gráficos

1 Introdução

1.1 Objectivos

No presente trabalho propôs-se a construção de um sistema servomecanismo didáctico envolvendo motores DC e Passo-a-Passo, com cargas dinâmicas.

Os objectivos iniciais para este trabalho eram: (i) confrontação com as dificuldades e aspectos práticos que envolvem o controlo de motores DC e passo-a-passo; (ii) o desenvolvimento de uma plataforma onde se possa analisar e actuar sobre motores DC e passo-a-passo, quando carregados com cargas rotativas dinâmicas; (iii) efectuar, de facto, as análises referidas no ponto anterior através da aplicação MATLAB (que dispõe da facilidade de utilização imediata de diversas ferramentas algorítmicas de tratamento de sinais); (iv) implementar e testar diversos algoritmos de compensadores de controlo, mediante objectivos pré-definidos; (v) implementar os algoritmos desenvolvidos num microcontrolador, por forma a que o sistema possa ser, também, operado autonomamente, sem recurso a computadores; (vi) contribuir para o estabelecimento de um laboratório de controlo no DETI.

1.2 Enquadramento

Os motores DC e passo-a-passo são componentes fundamentais de diversos equipamento comerciais e industriais, tais como discos rígidos de computadores, impressoras, linhas de montagem de processos fabris, etc. A forte competitividade e a consequente tendência para a redução dos custos de desenvolvimento destes equipamentos têm elevado a exigência sobre o desempenho efectivo dos vários componentes constituintes, incluindo os motores. Assim, torna-se cada vez mais relevante o controlo que é efectuado sobre estes dispositivos electromecânicos, sendo as técnicas de controlo implementadas cada vez mais elaboradas.

Reconhecendo a importância que o estudo do comportamento dos motores tem na formação de engenheiros electrotécnicos, concebeu-se a ideia de implementar uma plataforma de hardware que permita aos alunos de Sistemas e Controlo do DETI ter um contacto directo com equipamento real, podendo estes desenvolver algoritmos de controlo específicos para o controlo de motores eléctricos, e ter, também, a possibilidade de os implementar e testar esses algoritmos. Foi neste contexto que surgiu o trabalho que deu suporte a esta dissertação. Pretendeu-se que deste trabalho resultasse uma plataforma de interface simples com um computador em que, através da aplicação MATLAB (que é a aplicação utilizada nas aulas de

Sistemas de Controlo para efectuar simulações de sistemas físicos), os alunos pudessem, de forma simples, implementar o código dos seus algoritmos de controlo.

Refere-se, ainda, que está em curso o processo de implementação de um Laboratório de Controlo do DETI que incluirá diversos equipamentos didácticos que suportarão as aulas das disciplinas da área de Controlo do DETI, assim como trabalhos de investigação nessa área. Pretendeu-se que a plataforma desenvolvida no trabalho aqui apresentado possa contribuir para o estabelecimento do Laboratório de Controlo do DETI, constituindo um dos equipamentos didácticos ao dispor dos alunos.

1.3 Resumo do trabalho desenvolvido

Foi desenvolvida uma plataforma de hardware que permite a análise e o controlo do comportamento de um motor quando carregado com uma carga dinâmica rotativa. Os motores em análise foram motores DC e Passo-a-Passo. No desenvolvimento da componente de hardware foi necessário projectar por blocos distintos os vários módulos, que são: a alimentação do sistema; a conversão digital-analógica; amplificação dos sinais que actuam sobre os motores; a medição de parâmetros de funcionamento do sistema (velocidade de rotação da carga e binário produzido pelo motor, através da medição da corrente do mesmo) com a respectiva conversão analógico-digital; e a unidade de processamento embutida. Foi também implementado um algoritmo de controlo, PID, no motor DC através da aplicação MATLAB e, posteriormente, na própria unidade de processamento da plataforma (um microcontrolador PIC). Todos estes módulos foram projectados e testados individualmente, sendo depois integrados numa plataforma única, que mantém uma ligação a um computador externo através de uma linha de comunicação série RS232.

A plataforma desenvolvida permite especificar o sinal que actua directamente sobre o motor em análise, assim como permite monitorar a sua velocidade de rotação e, também, a corrente por ele consumida, em cada instante de amostragem. Com estes dados é possível implementar uma malha fechada de controlo da velocidade de rotação do motor, podendo esta malha ser fechada no computador externo ou mesmo na própria unidade de processamento da plataforma.

Foram efectuados testes de desempenho do sistema, sendo os seus resultados apresentados nesta dissertação.

2 Descrição do Hardware

2.1 Fonte de alimentação

O objectivo principal de uma fonte de alimentação é transformar uma tensão disponível, normalmente a tensão da rede, numa tensão diferente. Como por exemplo transformar a tensão da rede, 220V AC, numa tensão 12V DC, que permita trabalhar com equipamentos de corrente contínua. Para muitos circuitos electrónicos a fonte de alimentação é parte integrante fundamental.

A fonte de alimentação pode ser vista como composta por quatro blocos, como mostra a figura 2.1-1.

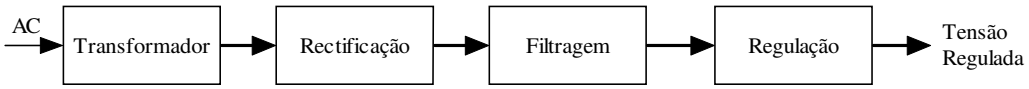


Figura 2.1-1 Diagrama de blocos da fonte de alimentação

Transformador – É um dispositivo destinado a transmitir energia eléctrica ou potência eléctrica de um circuito para outro, transformando tensões, correntes e ou de modificar os valores das impedâncias eléctrica de um circuito eléctrico.

Rectificação – Converte a tensão alternada numa tensão com componente DC.

Filtragem – Converte o sinal da rectificação numa tensão contínua.

Regulação – Regula a saída de modo a ter uma tensão DC constante.

2.1.1 Transformador

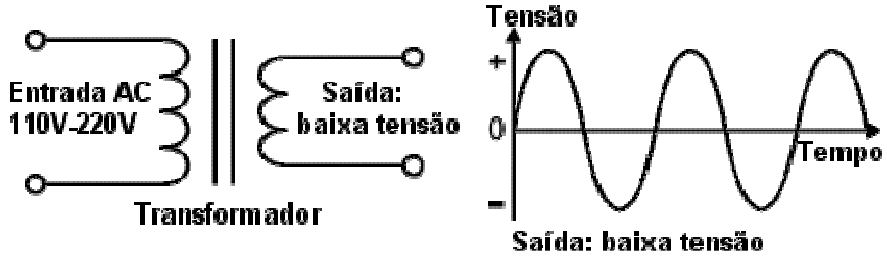


Figura 2.1-2 Transformador

O esquema do transformador e a forma de onda da tensão do secundário estão representados na figura 2.1-2. As relações entre o primário e o secundário encontram-se nas equações seguintes.

V_P = Tensão primário (entrada) V_S = Tensão no secundário (saída)

N_P = Número de espiras no primário N_S = Número de espiras no secundário

I_P = Corrente primário I_S = Corrente secundário

Tensão de pico teórico para a saída $V_p = V_S \times \sqrt{2}$.

Relação de transformação $= \frac{V_P}{V_S} = \frac{N_P}{N_S}$

Potência de saída = Potência de entrada $\Rightarrow V_S \times I_S = V_P \times I_P$

2.1.2 Transformador e Rectificação

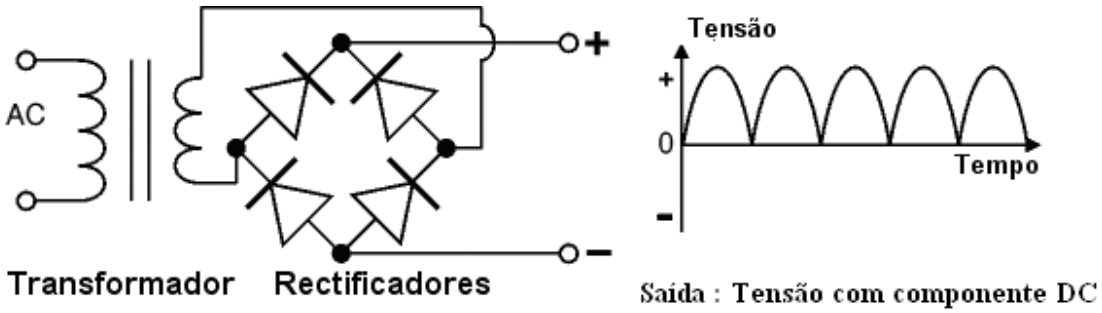


Figura 2.1-3 Transformador e rectificador

Existem várias formas de ligar os díodos de modo a criar um rectificador e converter um sinal AC num sinal periódico com componente DC. A ponte rectificadora produz uma rectificação de onda completa. Um rectificador de onda completa pode ser feito a partir de apenas dois díodos, com um transformador que tenha ponto médio, mas este método raramente é usado uma vez que os díodos são extremamente baratos, embora o transformador ficar muito mais dispendioso por ter dois enrolamentos iguais (só utiliza um em cada instante) e maior quantidade de ferro. Um único doído pode ser utilizado como rectificador mas só usa metade do sinal AC, logo, só produz meia-onda em DC.

Uma ponte rectificadora usa a onda AC (tanto a onda positiva como a negativa). 1,4V perde-se na ponte, porque cada um dos díodos rectificadores perde 0,7V (queda de tensão numa junção PN de silício) e na condução há sempre dois díodos em condução. As pontes rectificadoras são classificadas pela corrente máxima e a tensão inversa máxima que podem

suportar (esta deve ser, pelo menos, três vezes a tensão RMS de modo a que os rectificadores possam suportar os picos de tensão).

Um esquema usando uma ponte rectificadora com quatro díodos e a forma de onda resultante encontra-se demonstrado na figura 2.1-3.

2.1.3 Transformador com Rectificação e Filtragem

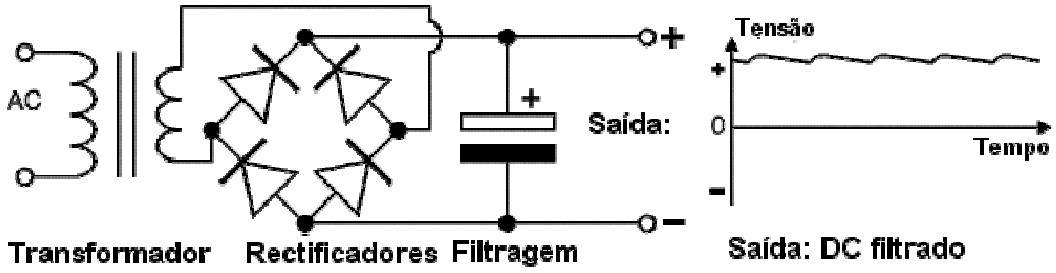


Figura 2.1-4 Transformado com rectificação e filtragem

Na figura 2.1-4 está representado um esquema eléctrico de um transformador com rectificação e filtragem e a respectiva tensão de saída após a filtragem. A filtragem é feita por um condensador electrolítico de grande capacidade ligado à saída DC. O condensador vai fornecer corrente para a saída quando a tensão varia no rectificador. Quanto maior for o valor do condensador menor será a variação da tensão DC resultante – a esta variação da tensão dá-se o nome de tensão de *ripple*. Para a maioria dos circuitos uma tensão de *ripple* na ordem dos 10% do valor da tensão pretendido é aceitável.

A figura 2.1-5 mostra a tensão não filtrada (linha pontilhada) e a tensão DC suavizada (linha sólida).

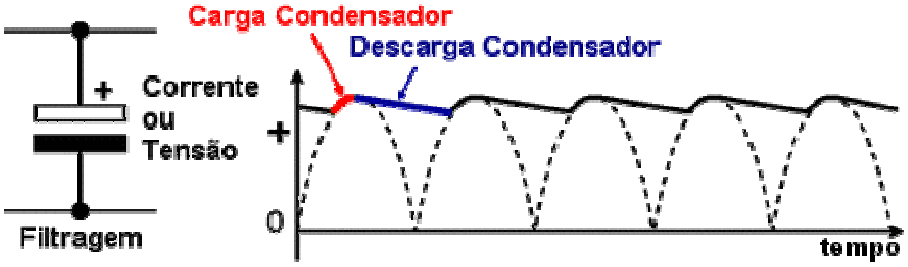


Figura 2.1-5 Carga e descarga no condensador

A tensão DC resultante da filtragem será aproximadamente o valor da tensão de pico antes da filtragem.

2.1.4 Transformador, Rectificação, Filtragem e Regulação

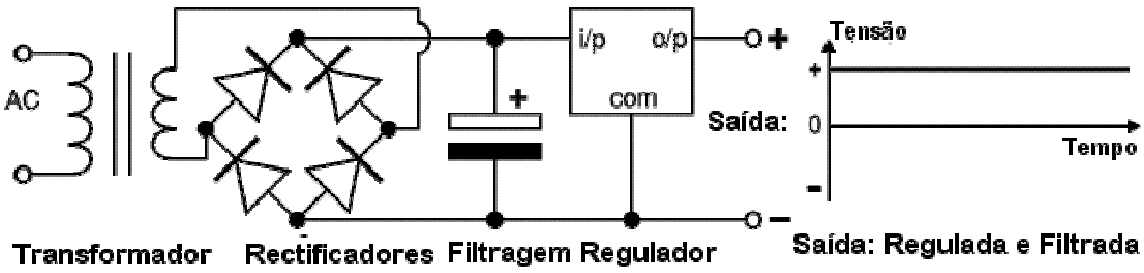


Figura 2.1-6 Transformador, rectificação, filtragem e regulação

Na figura 2.1-6 está representado o esquema completo de um bloco de alimentação e a respectiva forma de onda resultante. O regulador de tensão, converte a tensão de entrada numa tensão de saída constante. No mercado existem circuitos disponíveis para tensões positivas e negativas. Os números dos mais comuns começam com os números 78 ou 79 e terminar com dois dígitos que indicam a tensão de saída. O número 78 representa tensões positivas e o 79 tensões negativas, em que a tensão de entrada tem que ter o mesmo sinal que a tensão de saída.

As tensões de saída mais comuns nestes reguladores são: 5V, 6V, 8V, 12V, 15V, 18V e 24V, estando disponíveis para tensões positivas e havendo complementares para a tensão negativa.

A série MCT78XX e MCT79XX conseguem fornecer corrente até 1A. Para eliminar qualquer tensão AC de alta-frequência que possa aparecer na saída é recomendado adicionar um condensador em paralelo entre o comum e a saída do regulador. O valor desse condenador tem que ser superior a 0.1µF para não causar instabilidade do sistema. A tensão de entrada tem que ser superior à tensão de saída em pelo menos 2V. É aconselhável colocar um dissipador de calor no regulador para ajudar a dissipar o calor provocado de queda de tensão, principalmente se a tensão de saída for muito menor que a tensão de entrada. A maioria dos reguladores inclui protecção automática de excesso de corrente e de sobreaquecimento (protecção térmica). [9]

2.2 O motor DC

2.2.1 Introdução

O motor eléctrico é uma máquina destinada a transformar energia eléctrica em energia mecânica. Existem diversos tipos de motor eléctrico, sendo as classes mais vulgares as dos motores de corrente alternada (AC) e os de corrente contínua (DC). Neste trabalho não se abordaram motores AC pois, para o fim didáctico a que se destina a plataforma desenvolvida, achou-se que os motores DC seriam adequados aos objectivos de ensino, sendo inclusivamente mais simples lidar com a alimentação (e controlo) dos motores DC. Na figura 2.2-1 pode-se ver 3 motores DC.

Figura 2.2-1 exemplo de motores DC

2.2.2 Aspectos Construtivos

O motor de corrente contínua é composto de duas estruturas magnéticas: o estator (enrolamento de campo ou íman permanente) e o rotor (enrolamento de armadura).

O estator é composto de uma estrutura ferromagnética com pólos salientes nos quais são enroladas as bobines que formam o campo magnético (quando percorridas por corrente eléctrica), ou de um íman permanente.

O rotor é um electroíman constituído de um núcleo de ferro com enrolamentos na sua superfície que são alimentados por um sistema mecânico de comutação. Esse sistema de comutação é formado por um colector, solidário ao eixo do rotor, que possui uma superfície cilíndrica com diversas lâminas às quais são ligados os enrolamentos do rotor, e por escovas fixas que exercem pressão sobre o colector e que são ligadas aos terminais de alimentação.

Um motor de corrente eléctrica com um par de pólos é constituído por seis partes, um esquema deste motor pode-se ver na figura 2.2-2. As seis partes que o constituem são: a armadura (ou rotor), o colector, as escovas, o eixo, um campo magnético e uma fonte de alimentação. [2]

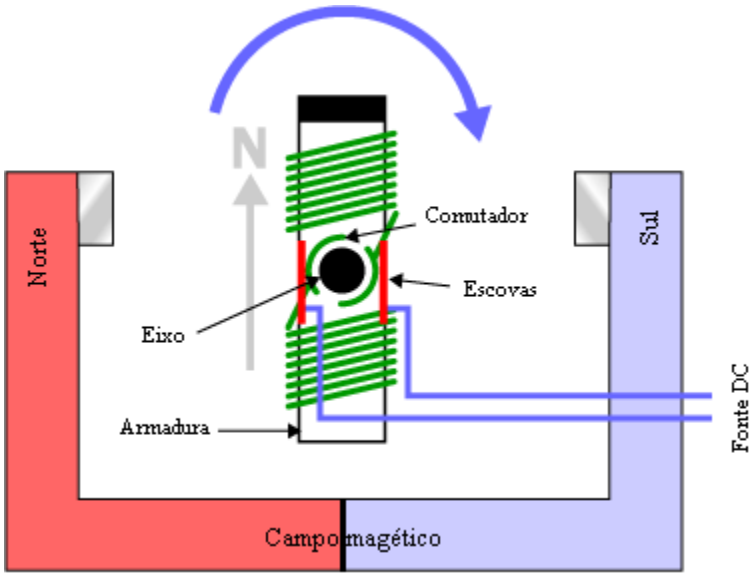


Figura 2.2-2 Modelo de um motor eléctrico simples

Nas figuras seguintes podemos ver um exemplo de um motor DC parecido com o usado neste trabalho. Nas figuras 2.2-3 e 2.2-4 podemos ver o corpo do motor (a parte metálica), o eixo, uma tampa de nylon e dois fios onde se pode ligar a alimentação do motor. Se se inverter os fios o motor irá rodar no sentido oposto.

Figura 2.2-3 Motor DC

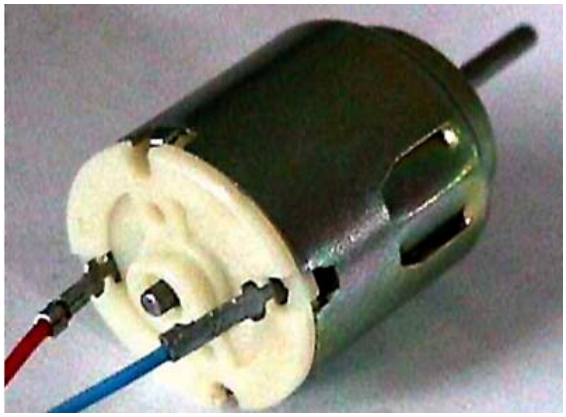


Figura 2.2-4 Outra vista do motor DC

Na figura 2.2-5 pode-se ver as escovas do motor que se encontram na parte de nylon.

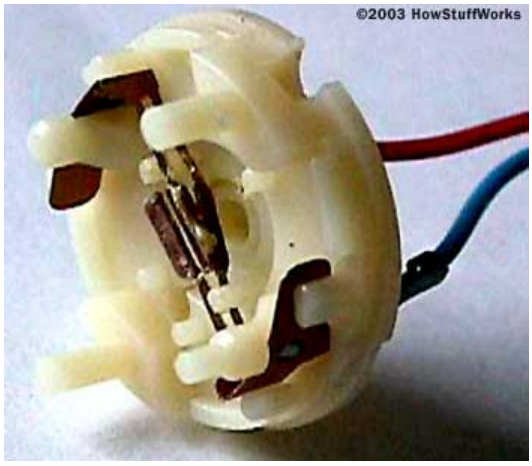


Figura 2.2-5 Escovas do motor

O eixo suporta a armadura e o colector. A armadura é um conjunto de electroímanes neste caso. A armadura neste motor é um conjunto de finas placas de metal empilhadas, com um fio de cobre fino enrolado em torno de cada um dos pólos da armadura. As extremidades de cada fio, um por cada pólo, são soldadas a um terminal, que depois está ligado ao colector. Na figura 2.2-6 podemos visualizar isto.

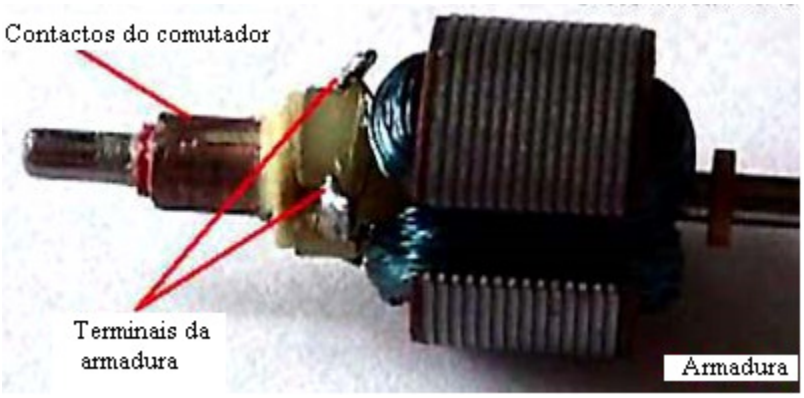


Figura 2.2-6 Interior de um motor

A parte final do motor eléctrico DC é o íman de campo. O íman deste motor é formado pelo próprio mais dois ímans permanentes curvos como se pode observar nas figuras 2.2-7 e 2.2-8.

Figura 2.2-7 Ímans permanentes no motor

Figura 2.2-8 ímans permanentes

2.2.3 Princípio de funcionamento

O princípio de funcionamento de todos os motores eléctricos é com base em ímans e magnetismo, um motor usa ímans para criar movimento. Pela lei fundamental do magnetismo, sabemos que pólos opostos se atraem e pólos iguais repelam-se. Portanto se se tiver duas barras de íman em que cada uma tenha uma extremidade denominada “sul” e a outra de denominada “norte”, isto fará com que a extremidade “norte” atrai a extremidade “sul” da outra barra, e do modo similar acontece para a extremidade “sul”. A extremidade “norte” repela a extremidade “norte” da outra barra, e do modo similar acontece para a extremidade “sul”. É devido a este princípio de atracão e repleção que o motor roda.

A figura 2.2-9 mostra, de maneira simplificada, o funcionamento do motor DC de dois pólos.

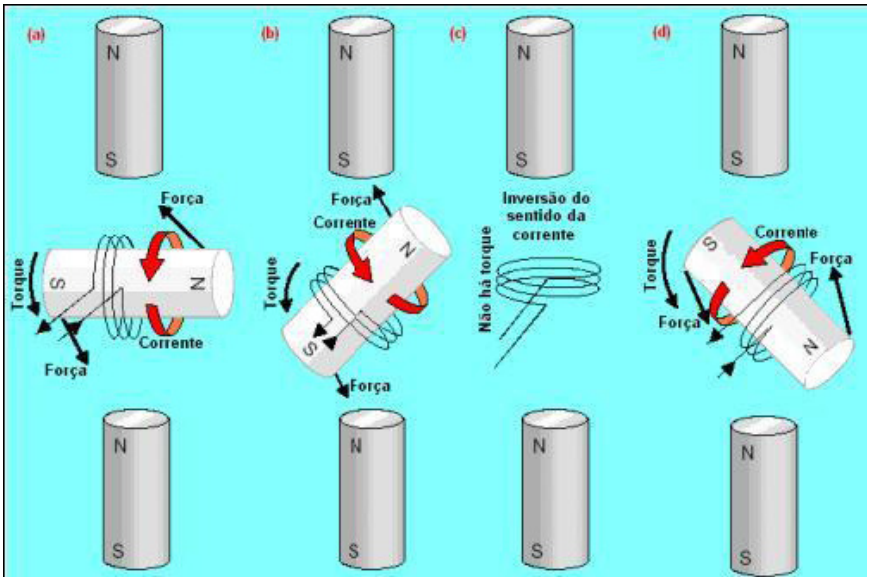


Figura 2.2-9 Diagrama de funcionamento de um motor DC com íman permanente

A figura 2.2-9 é um desenho esquemático simples de um motor onde o estator é constituído por ímans permanentes e o rotor é uma bobine de fio de cobre esmaltado por onde circula uma corrente eléctrica. Uma vez que as correntes eléctricas produzem campos magnéticos, essa bobine comporta-se como um íman permanente, com seus pólos N (norte) e S (sul) como mostrados na figura.

Na situação ilustrada em (**a**) onde a bobine está na horizontal, a bobine sofre um binário que age no sentido de rodar a bobine no sentido anti-horário, porque os pólos opostos se atraem. A bobine sofre aceleração angular e continua o seu movimento para a esquerda, como se ilustra em (**b**). Esse binário continua até que os pólos da bobine alcancem os pólos opostos dos ímans fixos (estator). Nessa situação (**c**), a bobine rodou 90º e não há binário, uma vez que os braços de alavanca são nulos (a direcção das forças passa pelo centro de rotação). O rotor está em equilíbrio (força e binário resultantes nulos). Esse é o instante adequado para inverter o sentido da corrente na bobine. Agora os pólos de mesmo nome estão muito próximos e a força de repulsão é intensa. Devido à inércia do rotor e como a bobine já apresenta um momento angular para a esquerda, esta continua a rodar no sentido anti-horário e o novo binário, agora gerado por forças de repulsão, situação em (**d**), colabora para a manutenção e aceleração do movimento de rotação.

Mesmo após a bobine ter rodado 180º, o movimento continua, a bobine atinge a posição vertical, rotação de 270º –, o binário novamente anula-se, a corrente novamente inverte seu sentido, há um novo binário e a bobine chega novamente à situação em (**a**) – volta de 360º, e o ciclo repete-se.

Essas atracções e repulsões bem coordenadas são o que fazem o rotor rodar. A inversão do sentido da corrente (comutação), no momento oportuno, é condição indispensável para a manutenção dos binários favoráveis, os quais garantem o funcionamento dos motores. A comutação consiste na mudança de uma lâmina do colector, onde as bobines são ligadas em série, para a próxima.

A figura 2.2-10 mostra um desenho esquemático simplificado de um motor de corrente contínua com apenas uma bobine, o colector e as escovas.

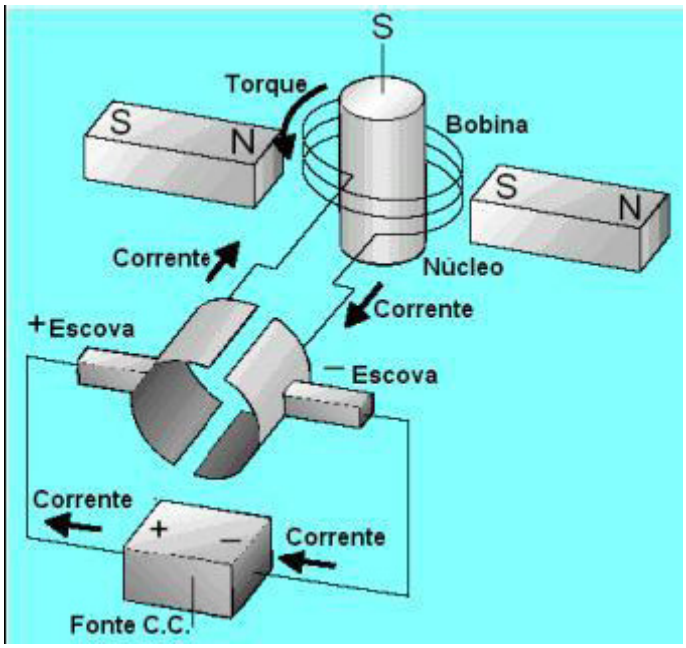


Figura 2.2-10 Colector e escovas

Na sua forma mais simples, o colector apresenta duas placas de cobre encurvadas e fixadas (isoladamente) no eixo do rotor; os terminais do enrolamento da bobine são soldados nessas placas. A corrente eléctrica chega por uma das escovas (+), entra pela placa do colector, passa pela bobine do rotor, sai pela outra placa do colector e retorna à fonte pela outra escova (-). Nessa etapa o rotor realiza sua primeira meia-volta, nessa meia-volta, as placas do colector trocam seus contactos com as escovas e a corrente inverte seu sentido de percurso na bobine do rotor, e o motor continua rodando, sempre com o mesmo sentido de rotação. [3]

2.2.4 Controle de Velocidade nos Motores DC

A velocidade de um motor DC é directamente proporcional à tensão de alimentação, se se reduzir a tensão do motor para metade, a velocidade do motor também vai reduzir para metade.

A alternativa a alterar a tensão de alimentação do motor é variar a tensão média enviada ao motor. Este método pode ser feito através de desligar e ligar a tensão de forma tão rápida que não se reflicta no motor. Este apenas vai reflectir a tensão média que lhe é aplicada. Por exemplo se se aplicar 12 Volts ao motor o motor vê 12 Volts, e se se desligar a tensão o motor vê 0 Volts, se o tempo que se aplica 12 Volts for igual ao tempo que se retira a tensão o motor vê uma tensão media de 6 Volts e o motor vai rodar com metade da velocidade que rodaria com tensão aplicada de 12 Volts.

Esta comutação pode ser implementada com transístores bipolares ou MOSFETs de potência. Estes elementos de potência permitem fornecer correntes elevadas tendo como sinal de entrada um sinal como baixo valore de corrente. Este tema é abordado no capítulo 2.4.

O tempo que um motor leva a acelerar e a desacelerar depende da inércia do rotor, quanto mais pesado for o rotor maior inércia terá, e quanto ao binário e atrito do motor.

O gráfico da figura 2.2-11 representa a velocidade de um motor que está a ser ligado e desligado de uma forma relativamente lenta. A linha a rosa representa a velocidade ao longo do tempo, e a linha azul o valor da tensão que está a ser aplicada ao motor ao longo do tempo.

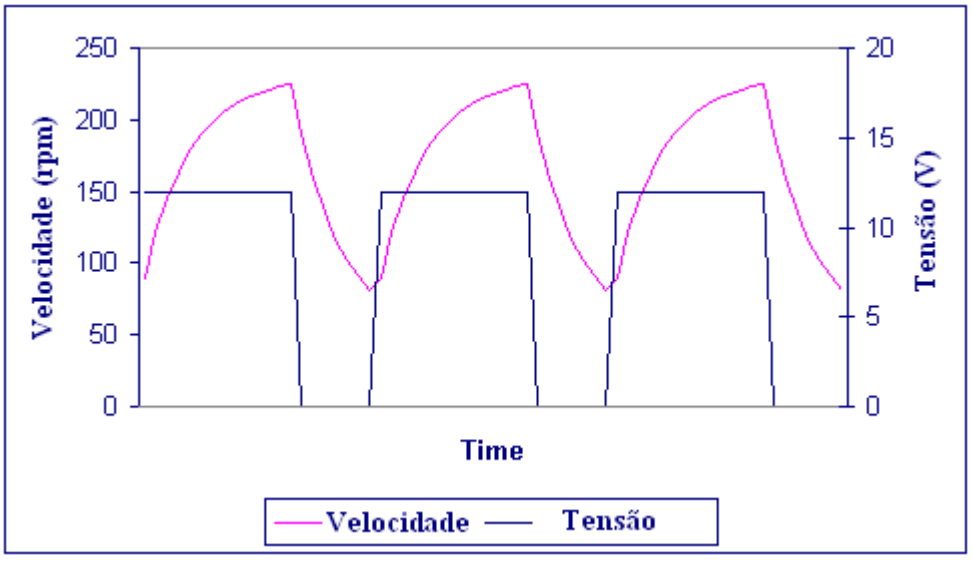


Figura 2.2-11Gráfico da velocidade de um motor

Pode-se observar que a velocidade média é cerca de 150 rpm, mas varia um pouco. Quanto maior for o tempo de comutação menor será a variação da velocidade, e a velocidade

será por isso mais estável. Este é o princípio de controlo de velocidade mais usado. Neste tipo de controlo a comutação do valor da tensão é denominado de PWM (pulso com modulação). [4]

Para o controlo do motor DC optou-se pelo controlo da velocidade através da variação da tensão aplicada ao motor, porque a abordagem do controlo por PWM iria ser usada para controlar o motor passo-a-passo, e em vez de se usar um circuito já existente do mercado como por exemplo uma ponte H, seria mais didáctico projectar um amplificador de potência para o controlo do motor DC.

2.3 O motor passo-a-passo

2.3.1 Introdução

Os motores passo-a-passo funcionam de forma diferente dos motores DC, que giram quando a tensão é aplicada a seus terminais. Os motores passo-a-passo são motores que podem ser controlados digitalmente. São usados em larga escala como por exemplo impressoras, plotters, scanners, drivers de disquetes, discos rígidos etc. Este tipo de motores são utilizados quando existe a necessidade de se ter precisão tanto em posição como em velocidade do rotor. Uma vantagem do motor de passo-a-passo em relação a outros motores é a estabilidade.

Um motor de passo-a-passo é um dispositivo digital, em que a informação digital é processada para conseguir um movimento controlado. Um impulso corresponde a um passo, N impulsos correspondem a N passos, isto se o motor estiver a funcionar em passo completo. Se o motor estiver a funcionar em meio passo é necessário dar Nx2 impulsos para dar N passos.

Precisão do motor passo-a-passo é determinada principalmente pelo número de passos por volta, quanto maior o número de passos, maior a precisão. Para precisão ainda maior, alguns controladores de motor de passo-a-passo dividem as etapas normais em meios passos.

Estes motores têm um número fixo de pólos magnéticos que determinam o número de passos por volta. Quanto maior for a velocidade de rotação do motor, menor será o seu binário.

As partes constituintes do motor de passo-a-passo são: o rotor (conjunto eixo – íman que rodam solidariamente na parte móvel do motor) figura 2.3-1 à direita, e o estator à esquerda.

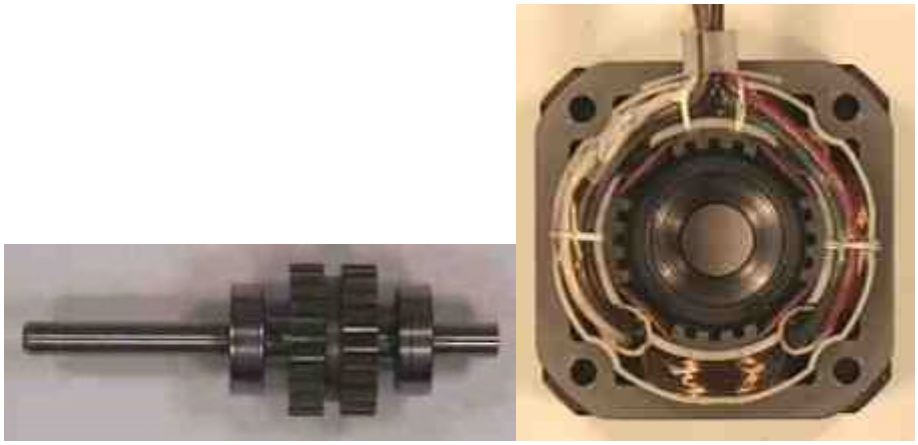


Figura 2.3-1Rotor e estator de um motor passo-a-passo

2.3.2 Valores importantes de um motor passo-a-passo

Graus por Passo, é sem dúvida a característica mais importante ao se escolher o motor, o número de graus por passo está directamente relacionado com o número de passos por volta.

Binário estático, força máxima que o rotor suporta quando está bloqueado, ou seja tensão aplicada aos enrolamentos mas sem comutação entre eles e sem perda de posição do rotor.

Binário dinâmico, neste caso é a força máxima obtida no rotor quando este está em movimento mas sem que exista perda de velocidade ou perda de passos, em regra geral os binários são apresentados em Newton x Metro [Nm].

Resposta de Passo é o tempo que o motor gasta para executar o comando.

Ressonância, o motor de passo a passo tem uma frequência natural. Quando o motor é submetido a uma frequência que corresponde á sua frequência de ressonância, este começa a oscilar e a perder passos.

Neste tipo de motores não é usual vir mencionado a tensão de trabalho, mas sim a corrente máxima admissível. A corrente aplicada ao motor é fundamental para a obtenção do binário pretendido. O controlo destes motores de modo geral têm como funcionamento manter a corrente constante, a tensão é que vai aumentando ou diminuindo consoante a frequência aplicada isto para que se consiga manter o binário o mais constante possível em todo o seu regime de funcionamento.

Um motor de passo é um actuador electromagnético rotativo, que converte mecanicamente impulsos digitais em rotação do eixo incremental. A rotação tem uma relação directa com o número de impulsos de entrada, e a sua velocidade está relacionado com a frequência dos impulsos.

Entre os passos, o motor mantém sua posição sem o auxílio de travões. Assim, um motor de passo pode ser controlado com precisão, ele roda uma distância proporcional ao número de impulsos enviados, e em seguida, para na posição esperada. Para escolher um motor de passo-a-passo é necessário ter em conta três critérios que são a velocidade e o movimento desejado e a carga que ele vai ter que mover.
Com a lógica apropriada, os motores de passo-a-passo podem ser bidireccionais, síncronos, proporcionar uma aceleração rápida, paragem e inversão e interage facilmente com outros dispositivos digitais.

2.3.3 Tipos de motores de passo mais usados

Os motores de passo podem ser encontrados em 2 tipos: íman permanente e relutância variável, existe também os motores híbridos, que são indistinguíveis do íman permanente ou relutância variável de ponto de vista de controlo. Motores de íman permanente possuem a tendência a dificultar o movimento quando se roda o seu eixo com os dedos (com o motor desligado) e os motores de relutância variáveis rodam livremente.

2.3.3.1 Motores Unipolares

Motores de passo-a-passo, tanto magnético permanente quanto híbridos com 5 ou 6 fios são geralmente esquematizados como mostra a figura 2.3-2, com um fio central em cada um dos enrolamentos. Na prática, usualmente o fio central é ligado ao pólo positivo da fonte, e os dois finais de cada enrolamento são levados ao pólo negativo alternadamente para reverter a direcção do campo magnético proveniente dos enrolamentos.

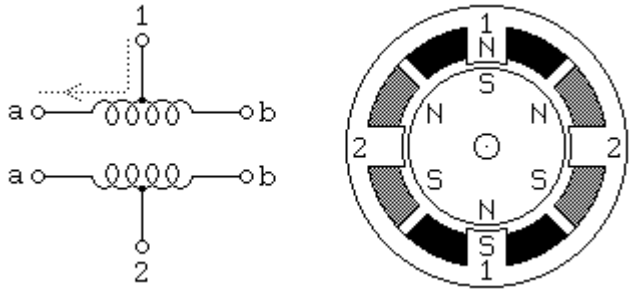


Figura 2.3-2Esquemático de um motor unipolar

A secção do motor mostrada na figura 2.3-2 é de 30º por passo, embora exista no mercado motores com diferentes passos este vai servir como exemplo.

O enrolamento 1 do motor é distribuído entre a parte de cima e a parte debaixo do pólo do estator, enquanto que o enrolamento 2 é distribuído entre a esquerda e a direita dos pólos do motor. O eixo é um íman permanente com seis pólos, três sul e três norte, colocados à volta do estator.

Conforme mostrado na figura 2.3-2, a corrente que flúi a partir do fio central do enrolamento 1 para um terminal faz com que o pólo superior do estator seja um pólo norte, enquanto o pólo inferior seja por isso um pólo sul. Isso atrai o rotor para a posição indicada. Se a corrente do enrolamento 1 for desligada e a do enrolamento 2 for ligada faz com que o rotor rode 30º, isto é um passo. Para fazer com que o motor continue a rodar, basta continuar a ligar e desligar os enrolamentos alternadamente. Uma sequência que faz com que este motor rode está representada na tabela 2.3-I, em que 1a, 1b, 2a, 2b são a corrente nos enrolamentos da figura 2.3-2. Assumindo como 1 o valor lógico positivo.

Tabela 2.3-I Sequencia de passo motor unipolar

Nº do passo	1ª	1b	2a	1b
1	1	0	0	0
2	0	0	1	0
3	0	1	0	0
4	0	0	0	1
5	1	0	0	0
6	0	0	1	0

Se quisermos que o motor funcione com meios passos a uma sequencia de alimentação das bobines que faz o motor rodar está na tabela 2.3-II.

Tabela 2.3-II Sequencia de passo motor unipolar em meios passos

Nº do passo	1ª	1b	2a	2b
1	1	0	0	0
2	1	0	1	0
3	0	0	1	0
4	0	1	1	0
5	0	1	0	0
6	0	1	0	1
7	0	0	0	1
8	1	0	0	1

2.3.3.2 Motores Bipolares

Motores de passo bipolares tanto com íman permanente quanto híbridos são construídos com exactamente os mesmo mecanismos usados nos motores unipolares, mas os dois enrolamentos são mais simples, sem fio central. Isto significa que, o motor é mais simples mas o circuito electrónico precisa controlar a inversão da corrente para cada enrolamento, isso torna-o muito mais complexo. O esquema da figura 2.3-3 mostra como o motor é configurado, enquanto a secção mostrada aqui é exactamente a mesma da secção da figura 2.3-2.

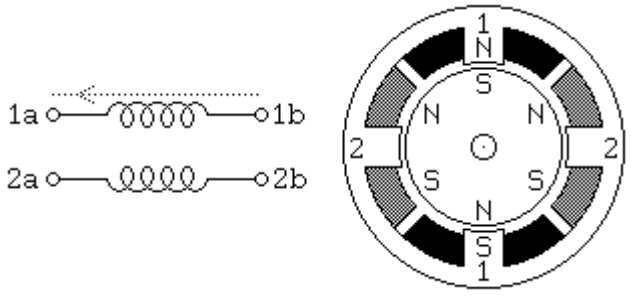


Figura 2.3-3 Esquemático de um motor bipolar

O circuito electrónico para estes tipos de motores é do tipo ponte H (pontes com 4 transístores). O motor precisa de uma ponte H para cada enrolamento. Basicamente, uma ponte H permite que a polaridade da energia aplicada em cada ponta de cada enrolamento seja controlada independentemente. A sequência de controle para um passo simples é mostrada na tabela2.3-III, em que os símbolos + e - para indicar a polaridade da energia aplicada em cada terminal do motor. Na tabela 2.3-III está representada a sequência para passo completo e na tabela 2.3-IV a sequencia com meios passos.

Tabela 2.3-III Sequência para rodar um motor com passo completo

Nº do passo	1a	1b	2a	2b
1	+	-	-	-
2	-	-	+	-
3	-	+	-	-
4	-	-	-	+
5	+	-	-	-

Tabela 2.3-IV Sequência para rodar um motor com meio passo

Nº do passo	1a	1b	2a	2b
1	+	-	-	-
2	+	-	+	-
3	-	-	+	-
4	-	+	+	-
5	-	+	-	-
6	-	+	-	+
7	-	-	-	+

8	+	-	-	+
9	+	-	-	-

2.3.3.3 Motores de relutância variável

O tipo mais comum de motores de passo de relutância variáveis é o motor possuir três enrolamentos, tipicamente conectados como mostra a figura 2.3-4, com um terminal comum para todos os enrolamentos. O fio comum normalmente vai para o pólo positivo da fonte e os enrolamentos são ligados em sequência.

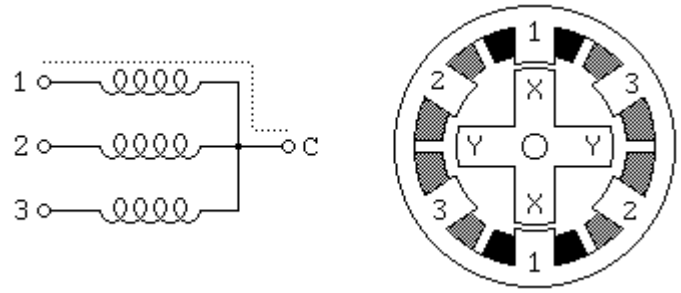


Figura 2.3-4Motor passo-a-passo com relutância variável

O motor exemplo da figura 2.3-4 é de 30° por passo. O eixo desse motor possui quatro dentes e os enrolamentos ficam em volta formando seis pólos enrolados em volta de dois pólos opostos. Com o enrolamento número 1 ligado, o dente do eixo marcado com X é atraído para o pólo desse enrolamento. Se a corrente através do enrolamento 1 for cortada e o enrolamento 2 for ligado, o motor irá rodar 30° (sentido horário) até que o pólo marcado com Y se alinhe com o pólo 2.

Para rodar esse motor de forma contínua, basta aplicar energia nos três enrolamentos em sequência. Usando lógica positiva, onde for 1 significa ter corrente através do enrolamento do motor, a sequência na tabela V irá rodar o motor ilustrado na figura 2.3-4 no sentido horário. Em que B1 B2 e B3 representam o enrolamento 1 2 e 3.

Tabela 2.3-V Sequencia de passos para rodar o motor

Nº do passo	B1	B2	B3
1	1	0	0
2	0	1	0
3	0	0	1
4	1	0	0
5	0	1	0
6	0	0	1
7	1	0	0
8	0	1	0

2.3.4 Estados e funcionamento de motores de passo

2.3.4.1 Três estados de um motor de passo

Desligado: Não há alimentação no motor. Não existe consumo de energia, e todas as bobines estão desligadas.

Parado: Pelo menos uma das bobines fica alimentada e o motor permanece estático num determinado sentido. Nesse caso há consumo de energia, mas o motor mantêm-se alinhado numa posição fixa.

Rodando: As bobines são alimentadas em intervalos de tempos determinados, impulsionando o motor a girar numa direcção.

2.3.4.2 Modos de operação de um motor de passo

Passo completo unipolar apenas uma bobine é alimentada a cada passo. Apresenta um menor binário, pouco consumo de energia e maior velocidade.

Meio passo a combinação de ter apenas uma bobine e em seguida ter duas bobines ligadas que gera um efeito de meio passo. Este consome mais energia que o passo completo mas é muito mais preciso. O binário é maior do que a do passo completo mas a velocidade é menor.

A forma com que o motor irá operar dependerá bastante do que se deseja controlar. Há casos em que o binário é mais importante, outros a precisão ou a velocidade. Essas são características gerais dos motores de passos. Ao trabalhar com motores de passos, precisamos saber algumas características de funcionamento como a tensão de alimentação, a máxima corrente eléctrica suportada nas bobines, o grau (precisão). As características mais importantes que devemos ter atenção para controlar um motor de passo são a tensão de alimentação e a corrente eléctrica que suas bobines suportam.

2.3.4.3 A velocidade de um motor de passo-a-passo

Para se controlar a velocidade de um motor passo-a-passo envia-se uma sequência de impulsos digitais num determinado intervalo de tempo. Quanto menor o intervalo de tempo, maior será a velocidade do motor. Para mudar o sentido de rotação do motor, simplesmente inverte-se a sequência dos passos descritos nas tabelas acima.

2.3.4.4 A precisão de um motor de passo-a-passo

Se se tiver um motor de passo com precisão de 7.5º, então para o motor dar uma volta, ter-se-á passos/volta = 360º / 7.5º = 48 passos por volta.

2.3.5 Drive de controlo motor de passo-a-passo

O *drive* (ou amplificador) do motor de passo-a-passo converte os sinais de comando em energia para alimentar as bobines do motor. Existem inúmeros tipos de drives, com diferentes funções, a escolha do *drive* depende do tipo de motor que estejamos a usar.

Para se actuar um motor precisamos de um hardware específico, as para simplificar a implementação podemos usar circuitos já existente no mercado.

2.4 Amplificador de potência para o motor DC

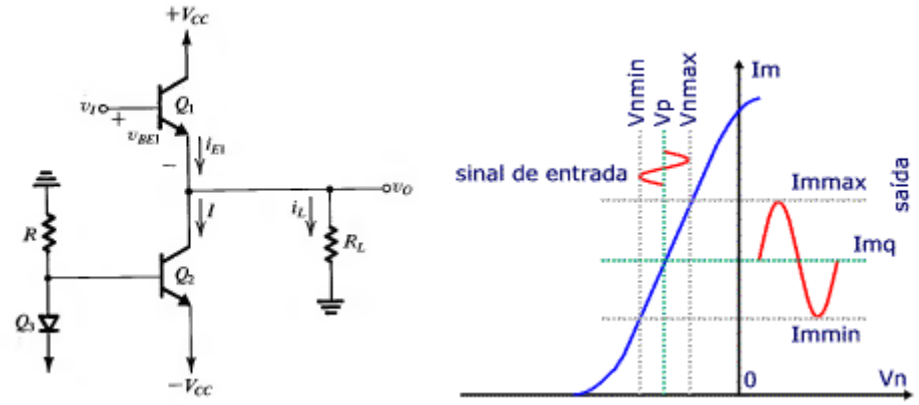
2.4.1 Introdução

Em grande parte das aplicações dos circuitos electrónicos existe a necessidade de amplificar sinais com um mínimo de distorção possível e o mais linear possível. Os amplificadores são projectados para elevar o nível de tensão de um sinal de entrada fraco. No entanto, a corrente de carga de um amplificador não é suficiente, em geral, para accionar cargas de baixa impedância, tipo um altifalante. Neste caso, após o amplificador, necessitamos de um amplificador de potência ou estágio de saída. A função do amplificador de potência não é gerar ganho de tensão, mas, sim, ganho de corrente para alimentar a carga. Os amplificadores usualmente são construídos à base de transístores bipolares ou FET's de potência. Existem quatro classes principais de estágios de saída. Essas classes são classificadas pelo período de condução do transístor de saída. [5] [6]

2.4.2 Classes do amplificador

2.4.2.1 Classe A

No amplificador classe A, o transístor de saída conduz durante os 360º do ciclo, 360º em termos de ângulo eléctrico, para um sinal sinusoidal. O rendimento é baixo, no funcionamento só com um transístor, o rendimento é teoricamente 25%, na prática é inferior a 25%, mas a qualidade é máxima, pois não existe transição entre dispositivos, sendo assim o sinal absolutamente ininterrupto. Pelo alto consumo, esta classe é usada quase exclusivamente por audiófilos. Um exemplo de um circuito em classe A está representado na figura 2.4-1. [5] [6] [7]

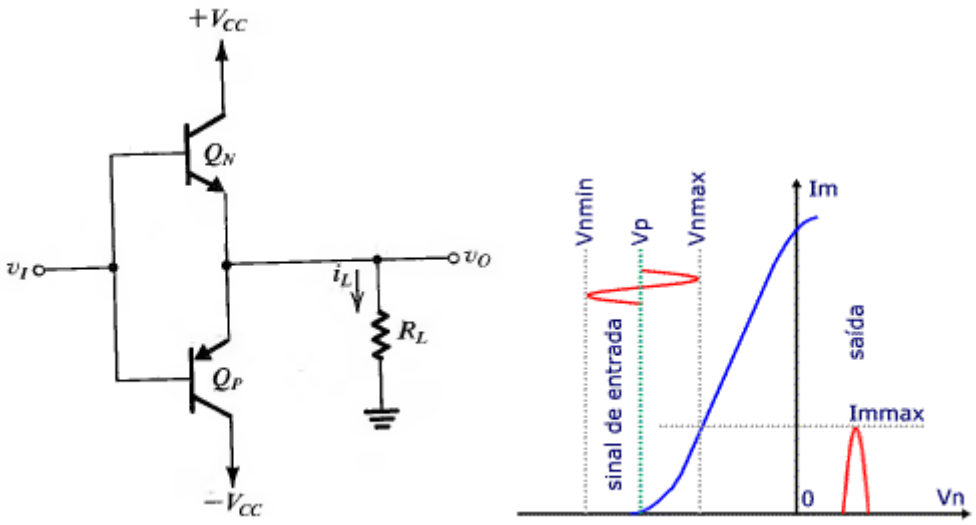


a) **b)**

Figura 2.4-1Exemplo classe A a) esquema eléctrico b) tensão no transístor de saída

2.4.2.2 Classe B

No amplificador classe B os transístores de saída apenas um dos transístores conduz em metade do ciclo, diz-se que o ângulo de condução é de 180º., a forma de onda da corrente é semelhante a uma sinusóide rectificada em meia onda. O rendimento máximo teórico é de 78,5%, mas como os circuitos não são ideais na prática o rendimento é sempre menor do que o rendimento teórico. Na figura 2.4-2 está representado o esquema eléctrico do classe B e as tensões de entrada e saída do amplificador.



a) **b)**

Figura 2.4-2 Classe B a) Esquema eléctrico b) tensão no transístor

Uma das características indesejáveis do classe B é a distorção de *crossover* devido à queda de tensão Vbe nos transístores, ver figura 2.4-3. De facto, quando o sinal de entrada é zero, e como as bases dos dois transístores estão ligadas entre elas, nenhum dos transístores está a conduzir. É necessária uma tensão base - emissor de cerca de 0.7V para que o *npn* comece a conduzir, e de –0.7V para que o *pnp* comece a conduzir. Este tipo de distorção é um problema sério, sobretudo para sinais pequenos. Há duas maneiras de o resolver, uma maneira é mediante a aplicação de realimentação no circuito, a outra é modificar o ponto de funcionamento dos transístores para que fiquem a funcionar em classe AB. Na figura 2.4-4 está o circuito de um classe B com realimentação para reduzir a distorção de *crossover*. Neste projecto optou-se por usar um classe B, com transístores de potência e com retroacção no circuito, para eliminar a distorção de crossover, porque em comparação com o esquema montado do classe AB este apresentou uma melhor estabilidade e a sua implementação é mais simples. Numa configuração em classe AB torna-se difícil que todos os componentes usados tenham o mesmo tipo de resposta face a variações de temperatura e que todos os transístores estejam também à mesma temperatura, porque torna-se difícil acoplá-los ao mesmo ponto. Como tal uma variação na temperatura pode fazer alterar significativamente a resposta do sistema.

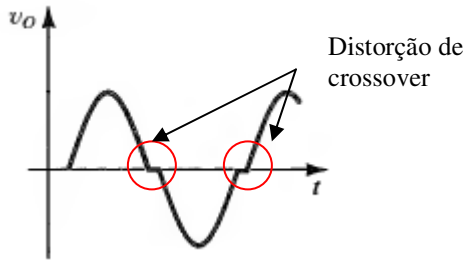


Figura 2.4-3 Distorção de *crossover*

A solução de realimentação não elimina por completo o *crossover* mas reduz substancialmente, a banda morta do amplificador que passa de ±0,7V a ±0,7/ A_0 em que A_0 é o ganho do amplificador. [5] [6] [7]

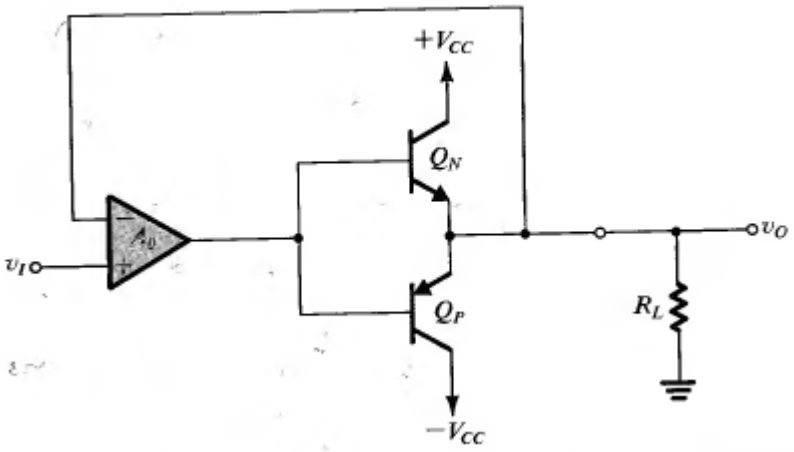
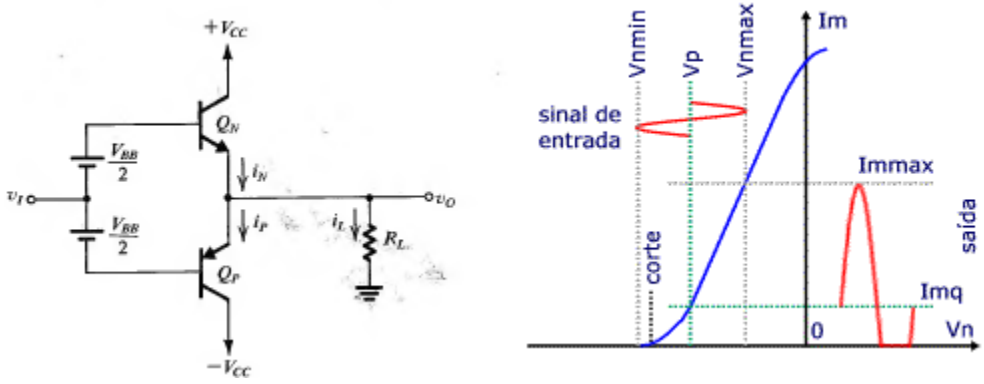


Figura 2.4-4 Circuito classe B com OPAMP realimentado para reduzir a distorção de *crossover*

2.4.2.3 Classe AB

No amplificador classe AB a condução ocorre entre 180º até 360º do ciclo de saída. Normalmente, o transístor de saída conduz apenas um pouco mais de 180º. O funcionamento em classe AB só difere do funcionamento em classe B por, em repouso, termos uma pequena corrente I a circular nos transístores. O esquema eléctrico e as formas das tensões estão representados na figura 2.4-5. [5] [6] [7]



a) **b)**

Figura 2.4-5 Classe AB a) esquema eléctrico b) tensão de entrada e saída

2.5 Encoder

O encoder é um dispositivo que converte o deslocamento linear ou giratório em sinais digitais. O *encoder* mais conhecido é o descodificador óptico, que consiste num disco que está montado no eixo do *encoder*, e que tem padrões que podem ser lidos através de um detector de luz que detecta a luz que está a ser emitida pela fonte luminosa, que está incorporado no próprio *encoder*. Na figura 2.5-1 está ilustrado um *encoder* com um disco com padrão e sensor de luz.

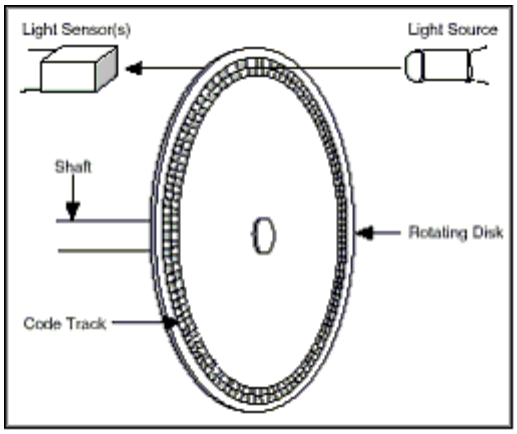


Figura 2.5-1Esquema de encoder

Enquanto o disco roda o sensor de luz vai detectar intervalos de luz resultantes da exposição do padrão do disco à luz emissora, resultando numa saída digital.

Um codificador incremental gera um impulso sempre que detecta luz, este tipo de codificador simples não possui posição absoluta, mas pode fornecer uma alta resolução a preço aceitável. A resolução do *encoder* é dada pelo número de buracos existentes no disco padrão.

Se um *encoder* só tiver um padrão no disco só temos um canal, com isto podemos através de um dispositivo apropriado medir a velocidade de rotação contando o número de impulsos por unidade de tempo, mas não é possível saber sem mais nenhum meio o sentido de rotação.

Para se poder saber o sentido de rotação é necessário ter pelo menos dois canais e dois detectores de luz. Quando se tem mais do que um padrão num anel estes devem estar desfasados e em quadratura, com isto podemos saber para que lado roda o *encoder* e também temos o acréscimo de ter uma maior resolução, mais buracos por volta implicam mais sinais digitais por volta. O resultado destes sinais estão em canais distintos, geralmente em *encoders* com dois canais estes canais são designados por canal A e B, e os padrões estão desfasados de 90º, ver figura 2.5-3. [8]

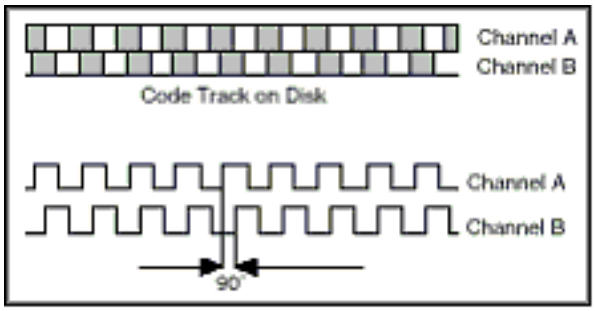


Figura 2.5-2 Padrão do *encoder* e saídas dos respectivos canais

3 Concepção da plataforma de controlo de motores

3.1 Diagrama do trabalho montado

De modo a simplificar o sistema optou-se por dividir o trabalho por blocos. Os blocos e respectiva designação encontra-se na figura 3.1-1. A descrição dos blocos e respectiva implementação encontra-se a seguir.

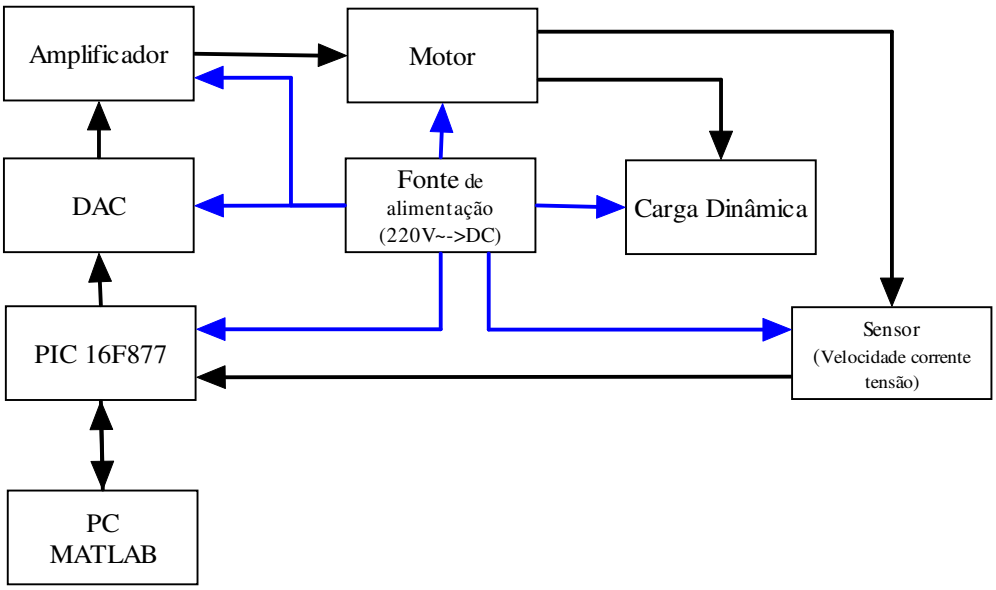


Figura 3.1-1 Diagrama de blocos da plataforma desenvolvida

Numa primeira abordagem a estrutura para ligar o motor DC ao passo-a-passo como encoder e a um motor que tivesse o efeito de carga encontra-se na figura 3.1-2. Esta solução apresentou alguns problemas, como o alinhamento dos veios não estava perfeito, causava muita vibração e não era um sistema muito flexível para alterar cargas e/ou motores.

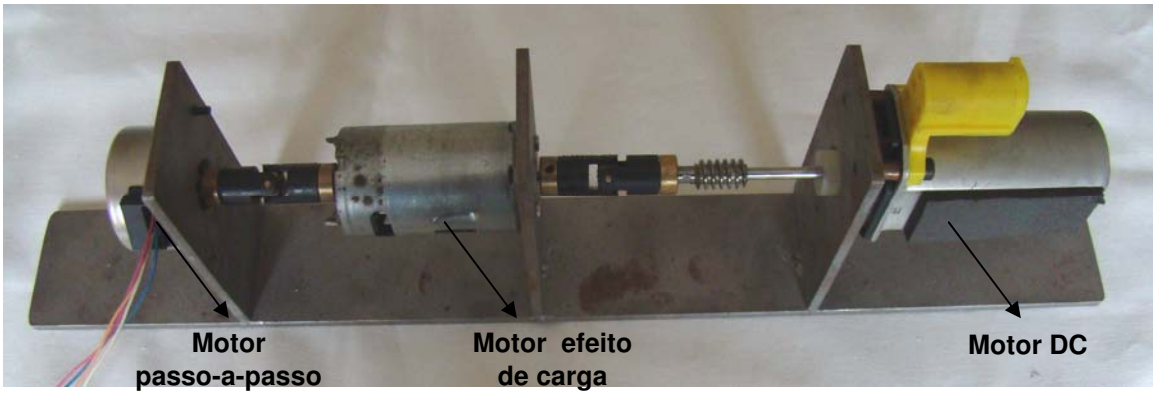


Figura 3.1-2 Ligação dos motores 1ª versão

Para ser mais fácil ligar os motores e poder alterar os efeitos de carga que estes podem estar sujeitos criou-se a segunda versão que se encontra na próxima figuras.

A placa com os motores ligados encontra-se na figura 3.1-3. Como se pode ver é possível através de correias ligar os motores a outros motores que podem fazer o efeito de carga. Também é possível aplicar mais peso ao veio do motor principal, as rodas de tracção que estão acopladas aos motores em estudo são de íman de modo ser mais fácil colocar peso para dificultar o movimento dos mesmos.

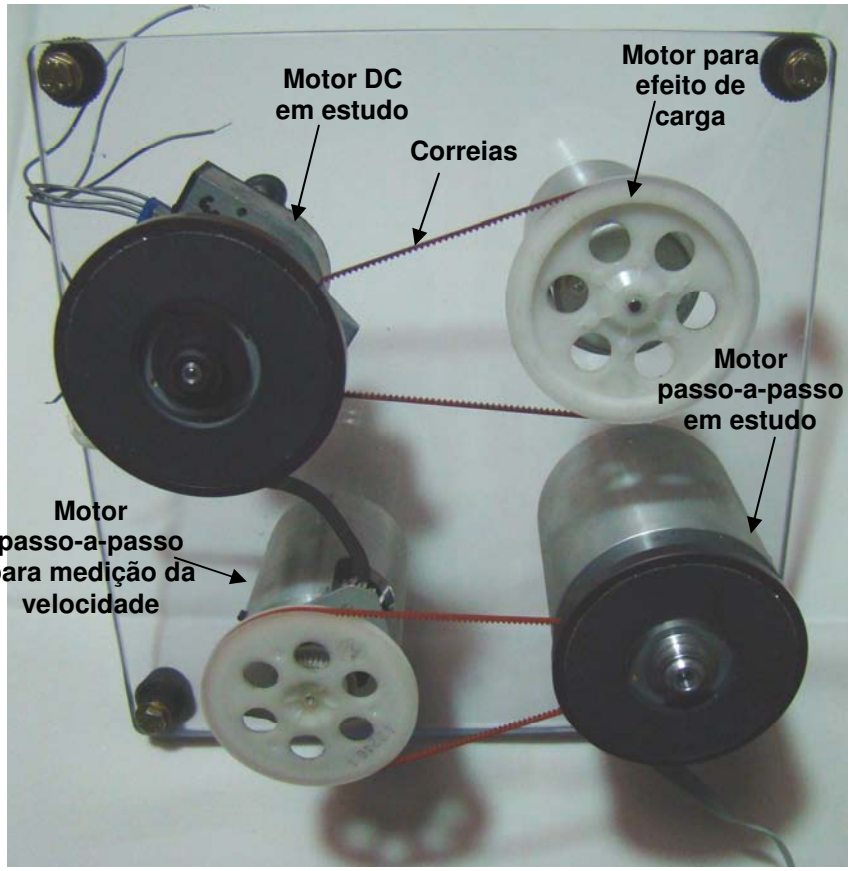


Figura 3.1-3 Placa com os motores

A vista geral do sistema criado está na figura 3.1-4. Onde se pode ver o sistema de acoplamento dos motores, as placas de controlo do sistema, a placa com a PIC e a porta de comunicação com o PC.

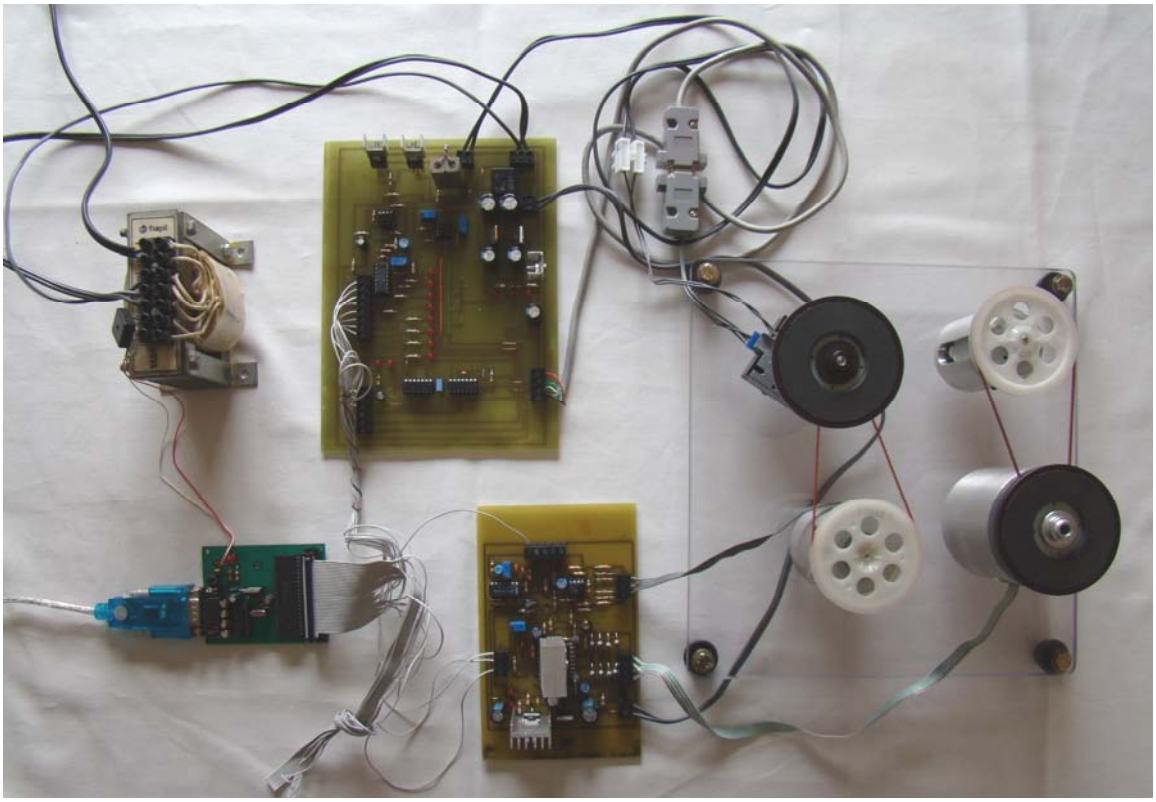


Figura 3.1-4 vista geral da plataforma

A placa que contem a alimentação do circuito e o controlo do motor DC encontra-se na figura 3.1-5. Nesta figura pode-se ver a PCB antes de ter os componentes soldados.

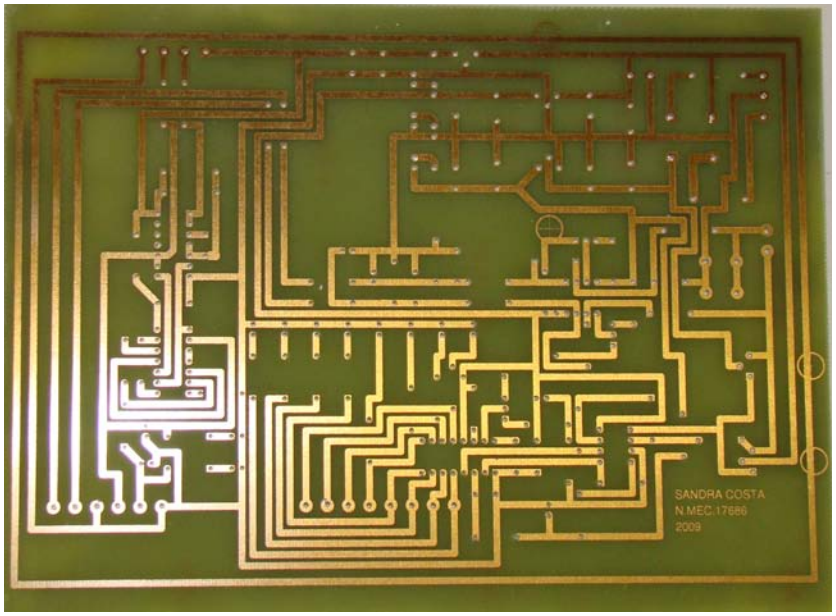


Figura 3.1-5 PCB do motor DC e alimentação

Na figura 3.1-6 pode-se ver a placa depois de montada. Como se pode observar na figura 3.1-6 existem leds que para controlo visual. Para tal temos um conjunto de 8 leds que estão ligados a saída do porto D, que corresponde a saída de alimentação em digital para o motor DC. Assim podemos ver se está ou não a ser enviado o que estamos à espera.

Também existem 3 leds que corresponde às tensões de alimentação do circuito, 5V, 12V e -12V respectivamente, e mais 2 que correspondem ao sentido de rotação do motor DC e à frequência de rotação.

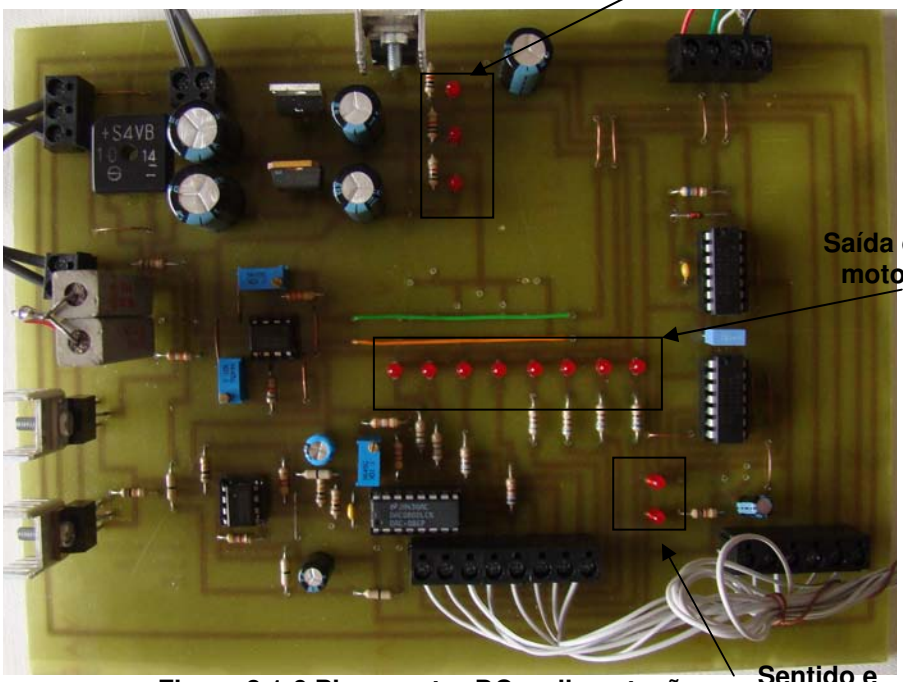


Figura 3.1-6 Placa motor DC e alimentações

Na figura 3.1-7 pode-se observar a placa de controlo do motor passo-a-passo. Nesta placa também existem leds de controlo visual. Temos um motor passo-a-passo a servir de encoder (medir a velocidade de rotação do motor que lhe está acoplado), na situação em que esta montado o circuito da figura 3.1-7 a leitura da velocidade é dada pelo circuito integrado que vai converter os impulsos num valor analógico que vai ser enviado à PIC, mas deixou-se como opção medir a velocidade de rotação como está o encoder que está acoplado ao motor DC, basta ligar os “canais A e B” deste motor no lugar onde estão ligados os canais A e B do encoder do motor DC.

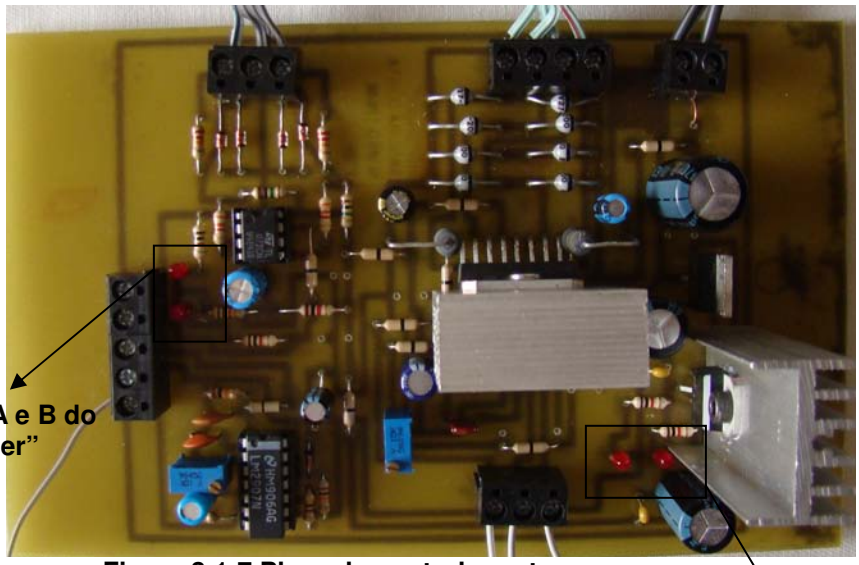


Figura 3.1-7 Placa de controlo motor passo-a-passo

3.2 Fonte de alimentação

O esquema da fonte de alimentação está na figura 3.2-1.

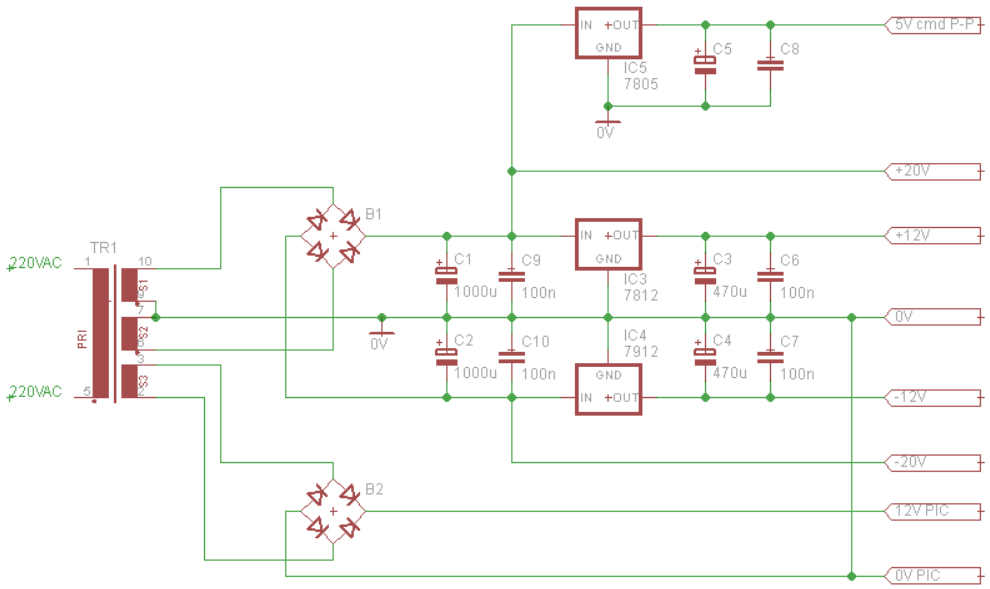


Figura 3.2-1 Esquema da fonte de alimentação

O transformador usado tem como especificações: tensão no primário 220V AC, as tensões disponíveis no secundário são 15V 0V 15V 0V e 10V, a potência do transformador é de 60W. A corrente máxima no secundário é $I_{0\max} = \frac{60}{15} = 4A$, se usarmos a tensão de secundário de 15V, e de $I_{o\max} = \frac{60}{10} = 6A$, para a tensão de secundário de 10V.

A PIC necessita de 5V DC para funcionar. Para obter esta tensão usou-se a tensão de 10V do secundário com uma ponte rectificadora e um regulador de tensão não comutado para 5V de tensão de saída. As tensões que vão alimentar o circuito que vai alimentar a PIC são as tensões de saída que se observam na figura 3.2-1 designadas por 12V PIC e 0V PIC. A tensão de pico que se obtém é dada por $V_p = 10 \times \sqrt{2} - 1.4 = 12.7V$.

As tensões que vão alimentar o circuito de potência e o motor são obtidas a partir da rectificação das tensões de secundário 15-0-15. As tensões de pico que se obtêm são dadas por $V_{Sr} = 15 \times \sqrt{2} - 1.4 = 19.8V$. Estas tensões são as tensões denominadas por 20V e -20V da figura 3.2-1.

As tensões que vão alimentar os blocos de controlo geral são dados pelas tensões anteriormente obtidas e depois utiliza-se dois reguladores de tensão não comutados, um para a componente positiva e outro para a componente negativa, para baixar a tensão de ±12V necessária para o funcionamento destes blocos. Estas tensões são denominadas por 12V e -12V na figura 3.2-1. A tensão 0V que aparece na figura é a tensão zero de referência que é usada por todos os bloco.

3.3 Conversor digital - analógico (DAC)

Um DAC, conversor digital - analógico, é um circuito electrónico que converte um valor digital num valor analógico. Neste projecto temos a tensão a ser aplicada ao motor que é enviada pela PIC. A PIC não tem saídas analógicas, sendo todas digitais. Por essa razão torna-se necessário converter esse valor digital numa tensão analógica que se possa aplicar ao motor.

Para o envio da tensão escolheu-se um porto da PIC que corresponde a 8 saídas, sendo o valor digital de 8 bits e varia entre 0 e 255. Para a conversão do sinal escolheu-se a DAC 0800.

Pretendia-se que os valores da tensão de saída variassem entre 10V e -10V. Após análise do datasheet do circuito integrado optou-se por ter uma saída que não fosse simétrica, para não ter dois valores digitais a corresponder a zero Volt, tendo em conta que a simetria dos valores não era um factor de relevo. Assim para um sinal tipo seno, em que a excursão do sinal pode

variar entre valores positivos e negativos, o sinal seria contínuo não havendo dor valores que iriam corresponder a zero.

Montou-se uma das configurações sugeridas no datasheet do circuito integrado, mas observou-se que quando se ligava o bloco de amplificação e o motor esta montagem apresentava valores que eram diferentes dos valores sem bolos adjacentes ligados. Um motivo para isso ocorrer era o facto de ao ligar os blocos seguintes a impedância de saída do circuito se alterar.

Então optou-se por fazer uma montagem que incluía um OPAMP na saída do circuito, com esta montagem verificou-se que a resposta do circuito já era praticamente a esperada. Uma das vantagens de se utilizar o OPAMP é de se eliminar o efeito de carga que se obtém quando se liga o circuito da DAC ao resto do circuito do trabalho o bloco de potencia e o motor. O esquema eléctrico desta montagem está na figura 3.3-1.

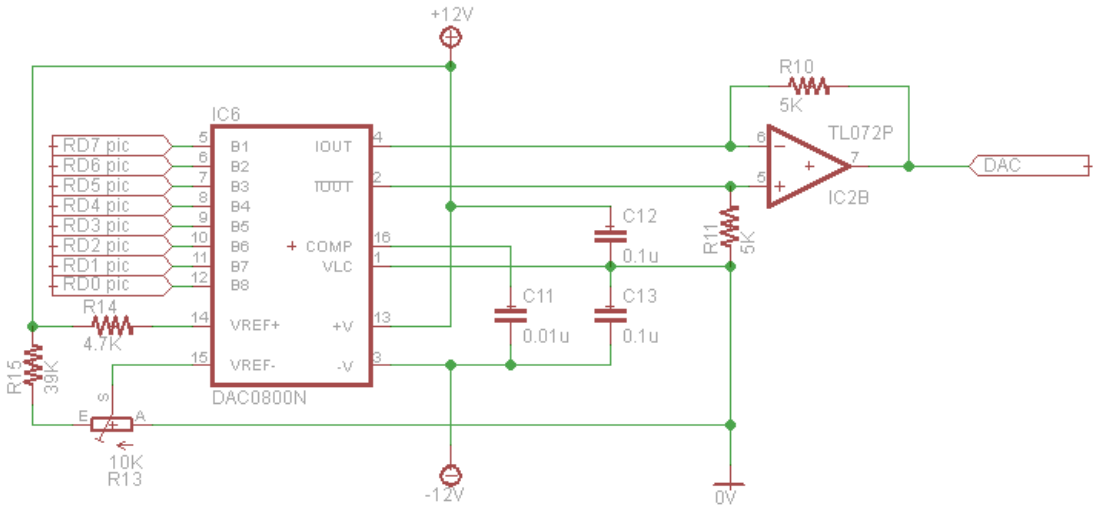


Figura 3.3-1 Esquema eléctrico da DAC

A tabela de conversão digital - analógico segundo o datasheet da DAC0800 está representada na tabela 3.3-I, em B1 representa o bit mais significativo e B8 o bit menos significativo.

Tabela 3.3-I Conversão digital – analógico

	B1	B2	B3	B4	B5	B6	B7	B8	E0
Pos. Full Scale	1	1	1	1	1	1	1	1	9,92
Pos. Full Scale - LSB	1	1	1	1	1	1	1	0	9,84
Zero Scale +LSB	1	0	0	0	0	0	0	1	0,08
Zero Scale	1	0	0	0	0	0	0	0	0,00
Zero Scale -LSB	0	1	1	1	1	1	1	1	-0,08
Neg. Full Scale + LSB	0	0	0	0	0	0	0	1	9,92
Neg. Full Scale	0	0	0	0	0	0	0	0	10,00

3.3.1 Resultados obtidos na montagem

Para se obter a resolução da DAC utilizou-se os dados da tabela 3.3-I e desenhou-se o gráfico para se ver qual a equação da recta de tendência. O valor obtido para a regulação foi de 0,0783 Volts, como se pode observar no gráfico 3.3-1.

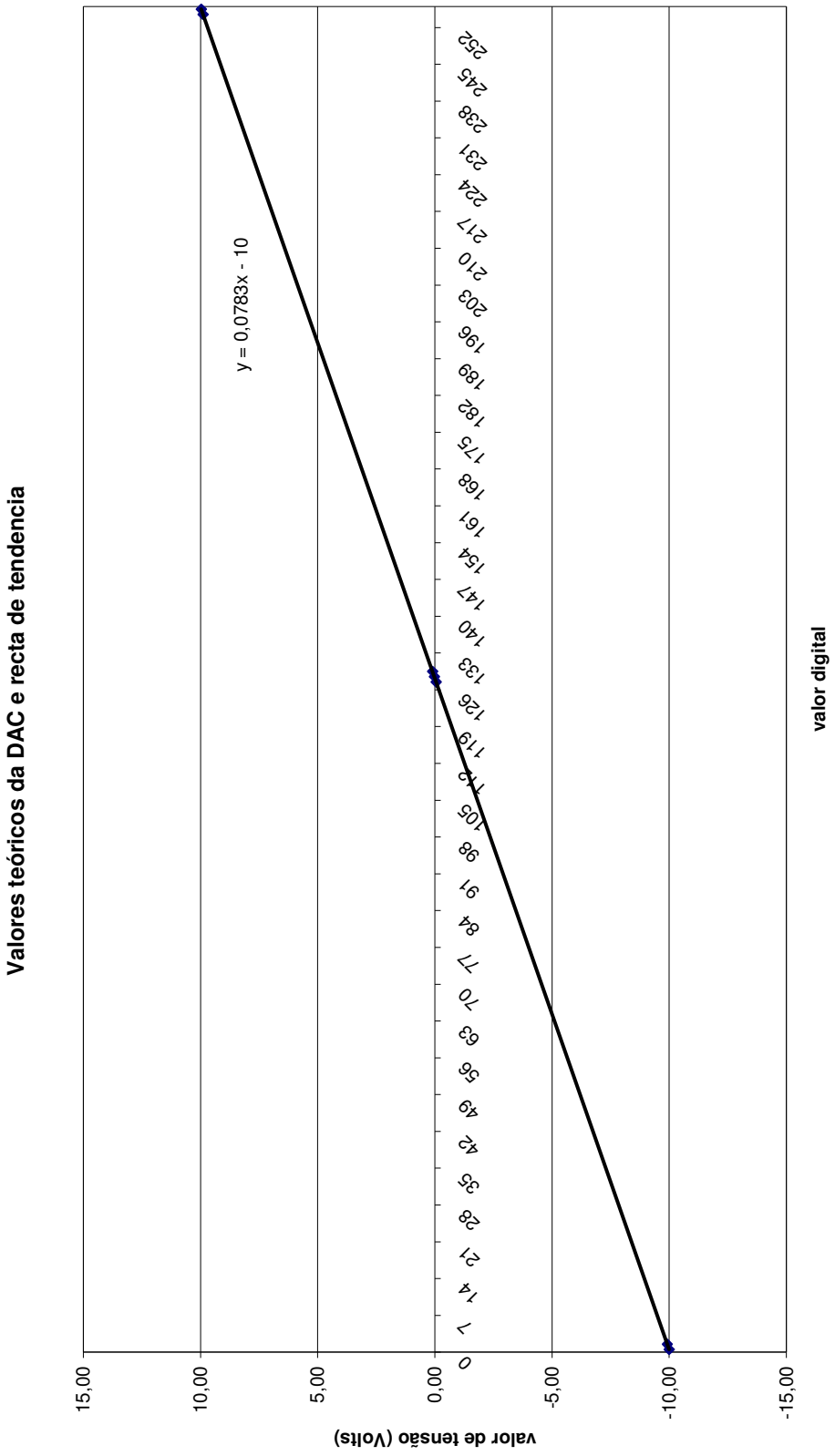


Gráfico 3.3-1 Valores teóricos

Título da Dissertação:

Sistema servo-mecanismo didáctico com motores DC e passo-a-passo.

Verificou-se o comportamento do circuito introduzindo vários valores digitais na entrada da DAC e medindo a saída que se obtinha na DAC em dois casos distintos. Primeiro sem nada ligado à saída do bloco e depois com o bloco ligado ao bloco de potencia com o motor ligado. Os resultados obtidos estão na tabela 3.3-II.

Tabela 3.3-II Valores obtido na conversão digital para analógico

Digital	Analógico			Digital	Analógico		
0-128	Teórico	Sem carga	Com motor	128-255	Teórico	Sem carga	Com motor
0	-10,00	-10,00	-9,98	128	0,02	0,02	0,02
1	-9,92	-9,92	-9,91	129	0,10	0,10	0,09
2	-9,84	-9,84	-9,83	130	0,18	0,17	0,17
3	-9,77	-9,77	-9,76	131	0,26	0,25	0,25
4	-9,69	-9,69	-9,68	132	0,34	0,33	0,33
5	-9,61	-9,61	-9,60	133	0,41	0,41	0,41
6	-9,53	-9,53	-9,53	134	0,49	0,49	0,48
7	-9,45	-9,46	-9,45	138	0,81	0,80	0,80
8	-9,37	-9,38	-9,37	148	1,59	1,58	1,58
18	-8,59	-8,60	-8,59	158	2,37	2,35	2,35
28	-7,81	-7,82	-7,81	168	3,15	3,14	3,14
38	-7,02	-7,04	-7,03	178	3,94	3,92	3,92
48	-6,24	-6,26	-6,25	188	4,72	4,70	4,70
58	-5,46	-5,48	-5,47	198	5,50	5,51	5,51
68	-4,68	-4,67	-4,67	208	6,29	6,29	6,29
78	-3,89	-3,89	-3,89	218	7,07	7,07	7,07
88	-3,11	-3,11	-3,11	228	7,85	7,85	7,85
98	-2,33	-2,33	-2,33	238	8,64	8,63	8,63
108	-1,54	-1,55	-1,55	248	9,42	9,41	9,41
118	-0,76	-0,77	-0,77	249	9,50	9,49	9,49
123	-0,37	-0,39	-0,39	250	9,58	9,57	9,57
124	-0,29	-0,31	-0,31	251	9,65	9,65	9,64
125	-0,21	-0,23	-0,23	252	9,73	9,72	9,72
126	-0,13	-0,15	-0,15	253	9,81	9,80	9,80
127	-0,06	-0,07	-0,07	254	9,89	9,88	9,88
128	0,02	0,02	0,02	255	9,97	9,96	9,95

Pela análise da tabela 3.3-II e dos gráficos 3.3-1 e 3.3-2 pode-se verificar que o sistema é praticamente linear, optou-se por fazer uma divisão dos valor, entre os valores que correspondem aos sinais negativos e aos positivos, para tornar o gráfico mais perceptível. Com a ligação do esquema de conversão ao circuito do amplificador do motor, os valores são praticamente iguais aos obtidos quando o circuito estava em aberto.

Nos gráficos 3.3-2 e 3.3-3, a linha azul corresponde ao valor analógico teórico, a linha vermelha os valores da tensão quando não se tem aplicado nada à saída da DAC e a linha verde corresponde à tensão com o circuito ligado ao amplificador com o motor ligado. Como se pode observar estas estão praticamente sobrepostas.

Nos gráficos 3.3-4 e 3.3-5 pode-se ver a diferença em módulo entre o valor teórico e o valor medido sem carga, nos gráficos 3.3-6 e 3.3-7 pode-se ver a diferença em módulo entre o valor teórico e o valor medido com carga, e nos gráficos 3.3-8 e 3.3-9 pode-se ver a diferença em módulo entre o valor medido sem carga e o valor medido sem carga. Pode-se concluir que o bloco da DAC está a funcionar correctamente.

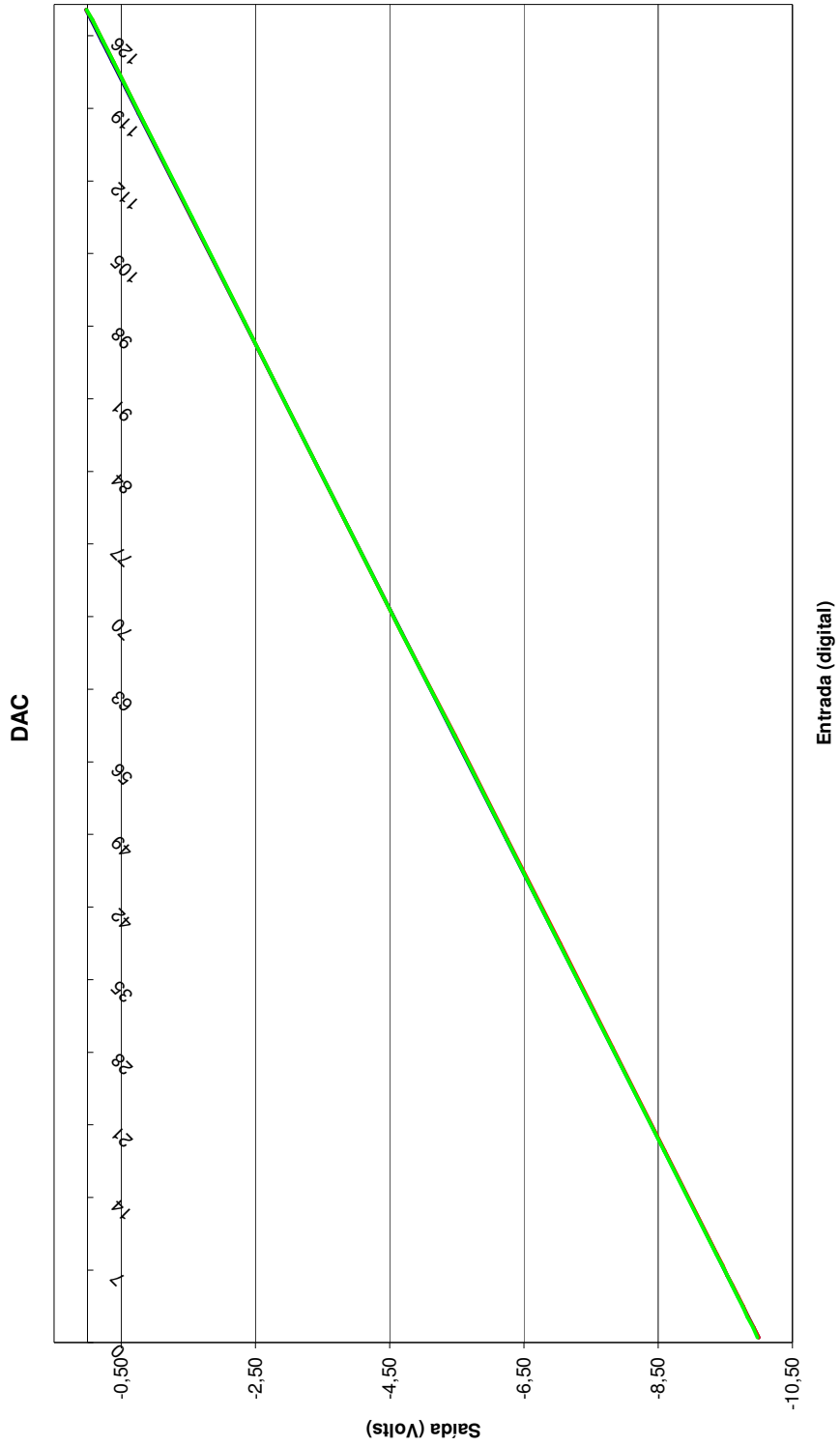


Gráfico 3.3-2 Tensão na saída da DAC para uma entrada a variar entre 0-128

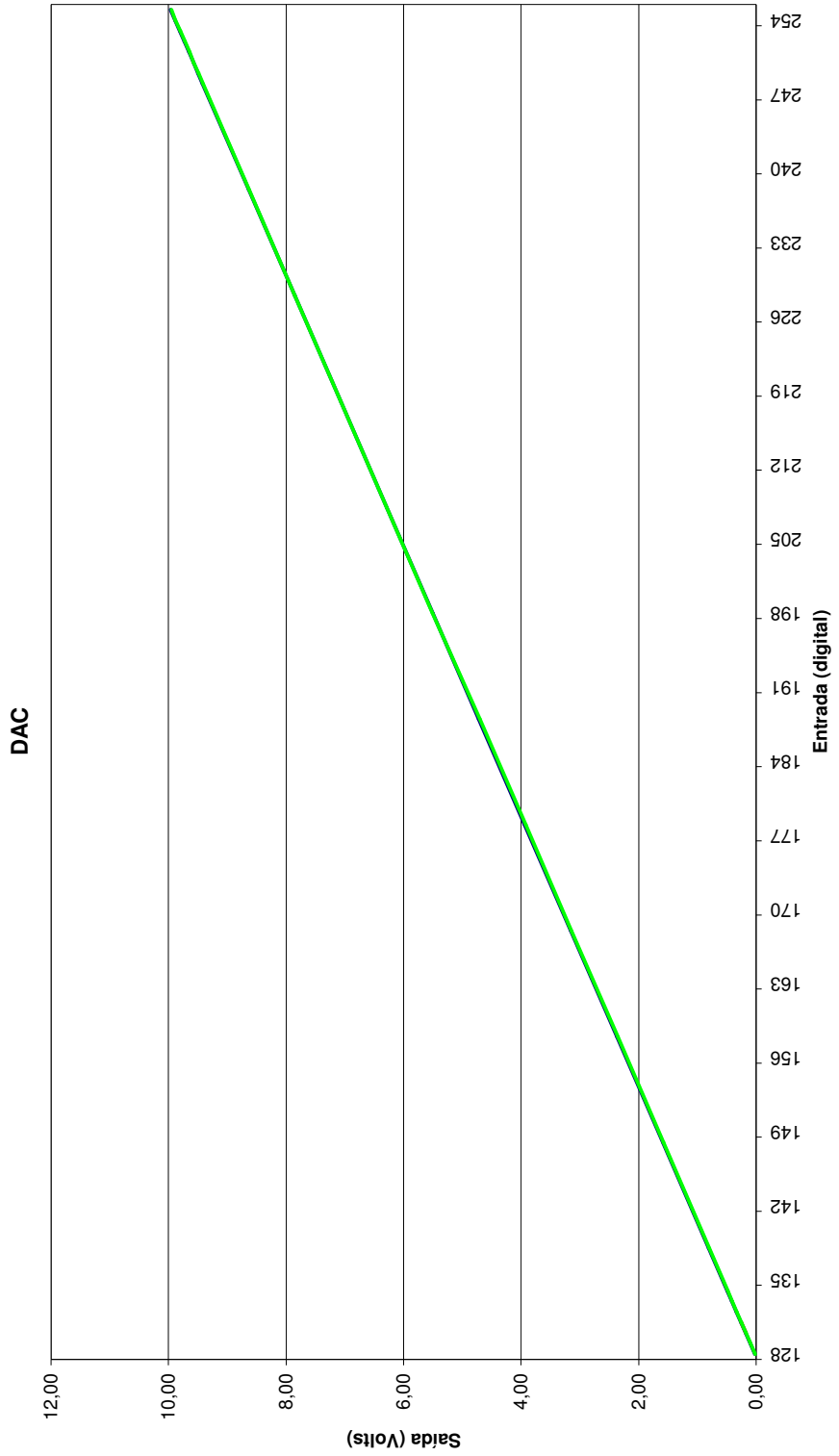


Gráfico 3.3-3 Tensão na saída da DAC para uma entrada a variar entre 128-255

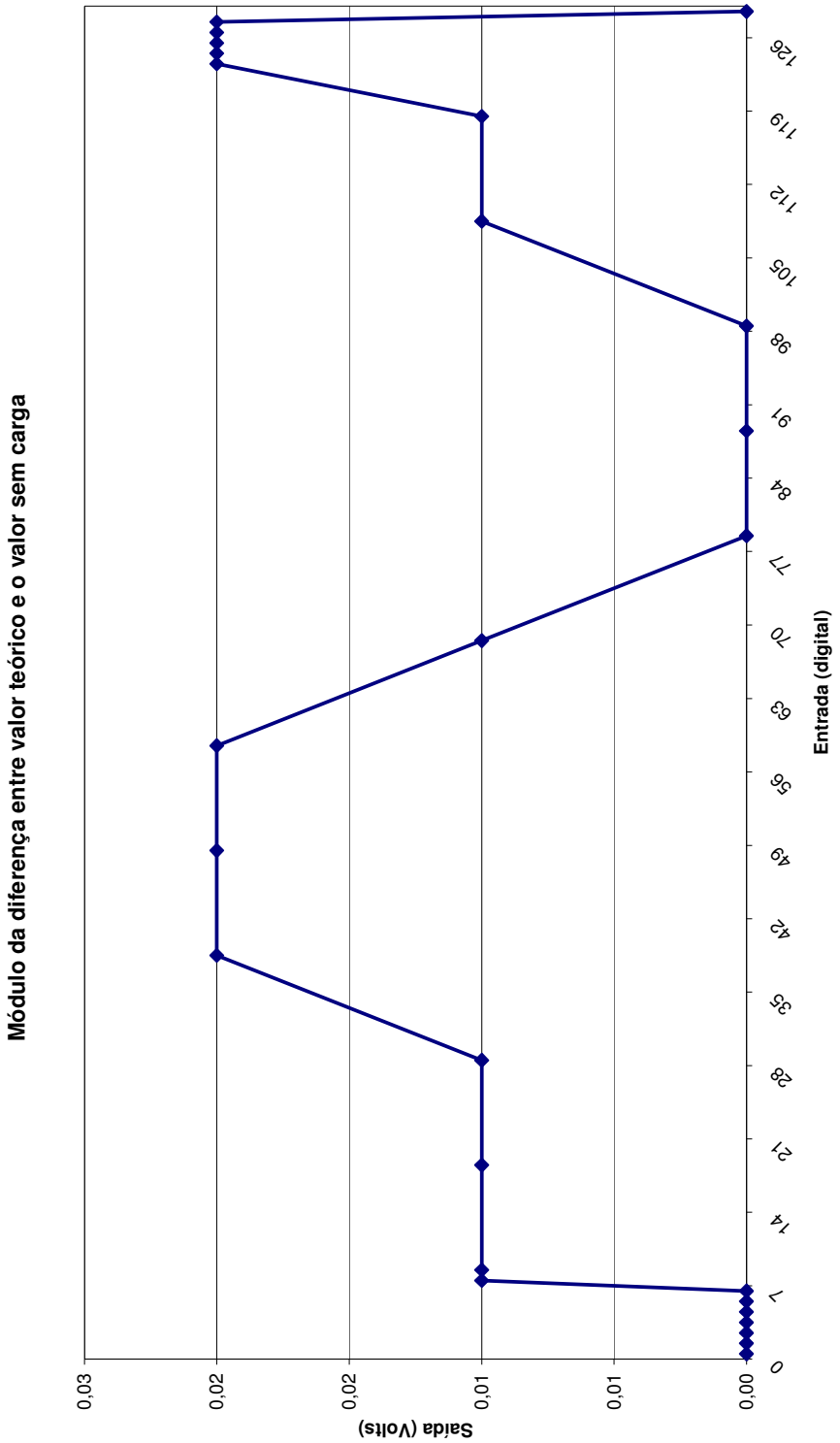


Gráfico 3.3-4 Diferença entre os valores esperados entre o valor teórico e o valor sem carga entre 0 e 128

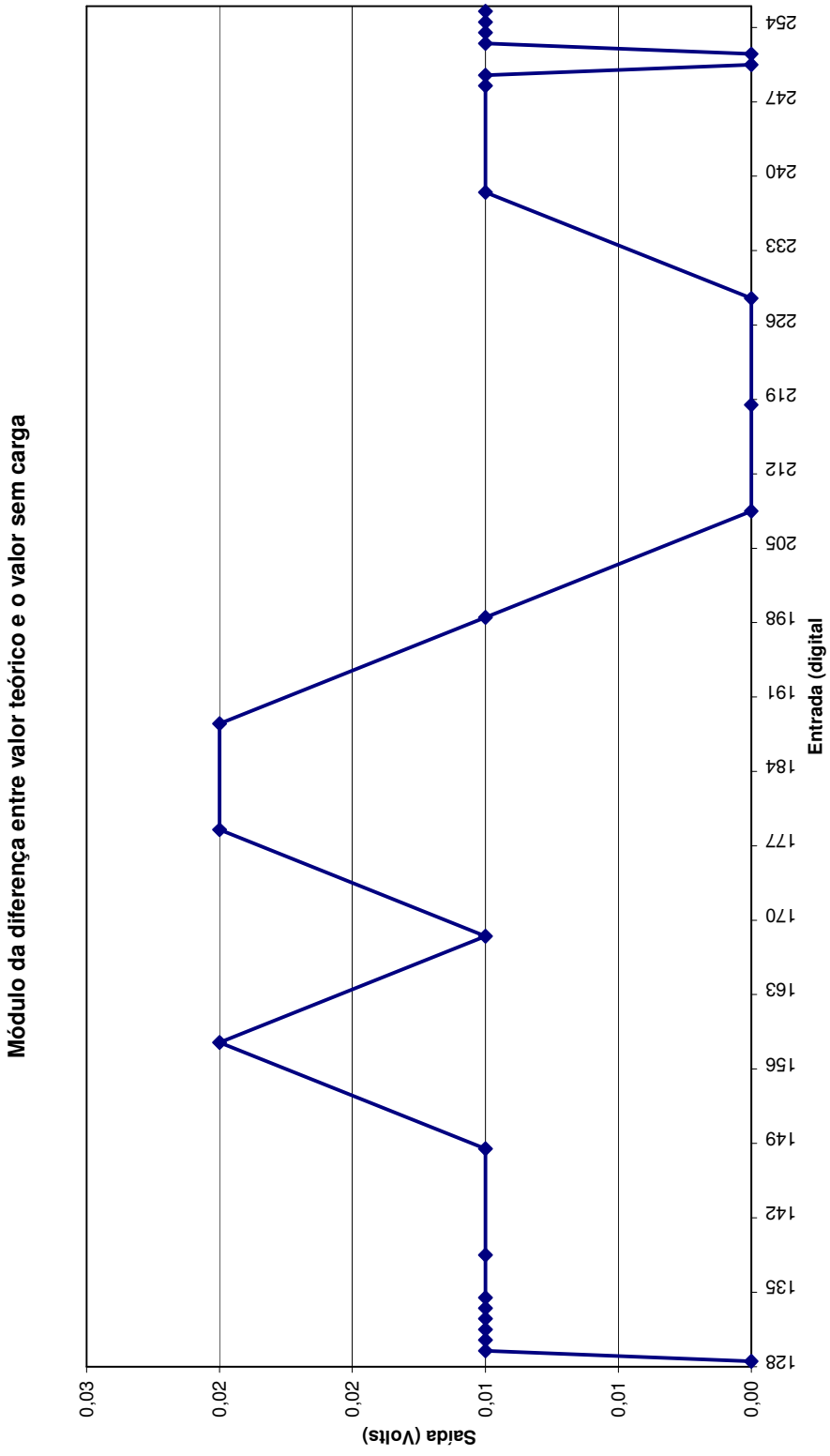


Gráfico 3.3-5 Diferença entre os valores esperados entre o valor teórico e o valor sem carga entre 128 e 255

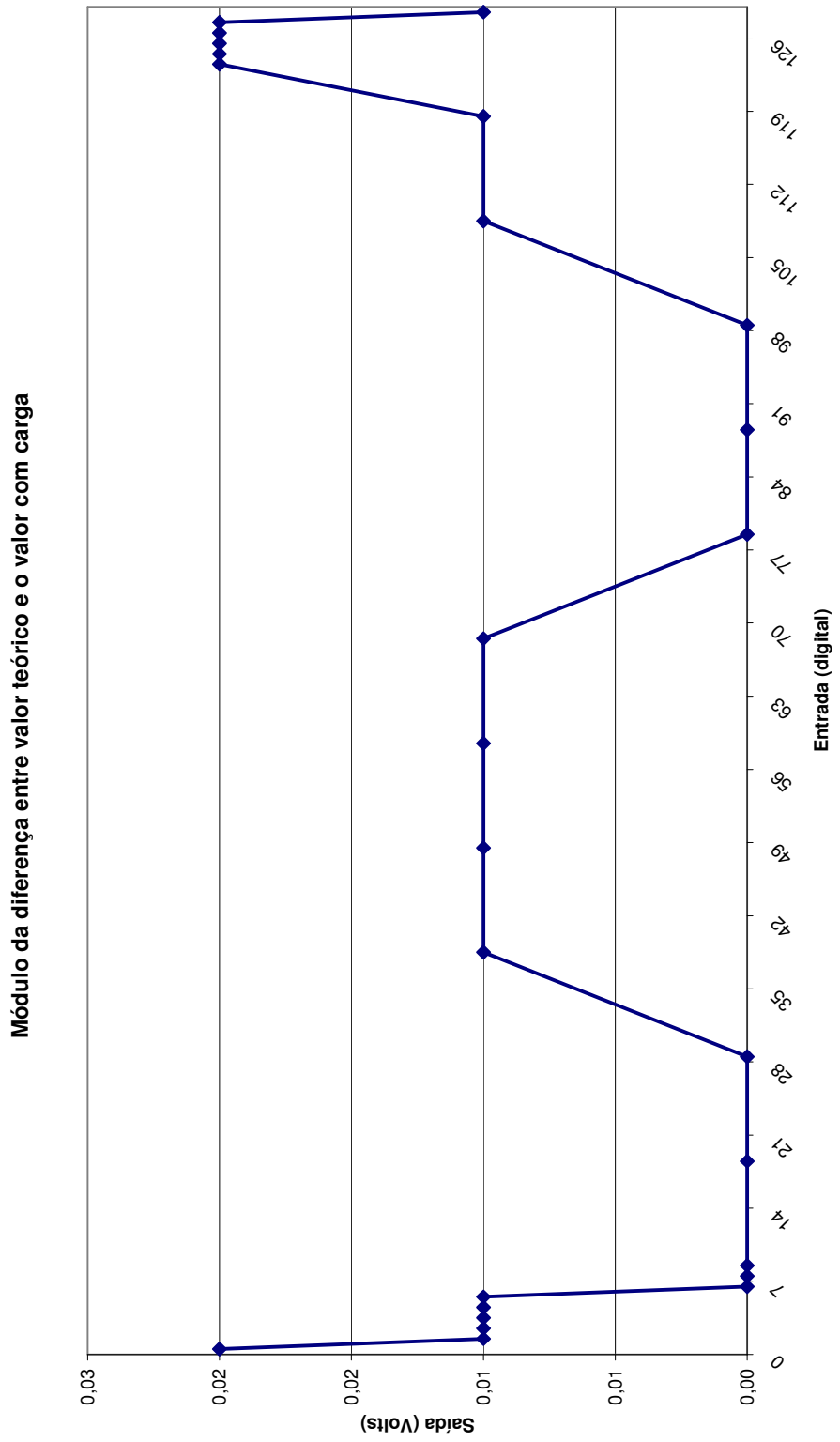


Gráfico 3.3-6 Diferença entre os valores esperados entre o valor teórico e o valor com carga entre 0 e 128

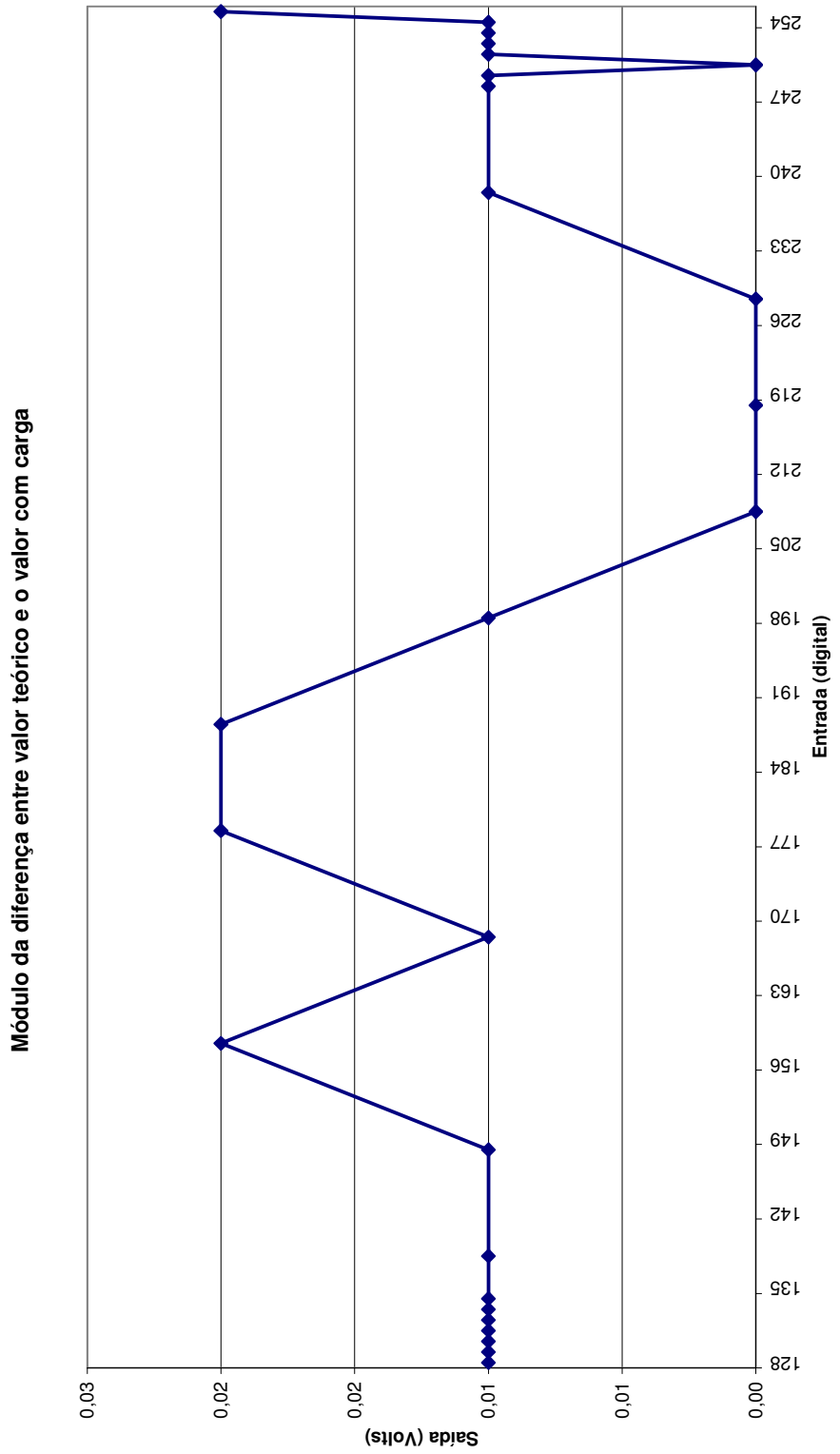


Gráfico 3.3-7 Diferença entre os valores esperados entre o valor teórico e o valor com carga entre128 e 255

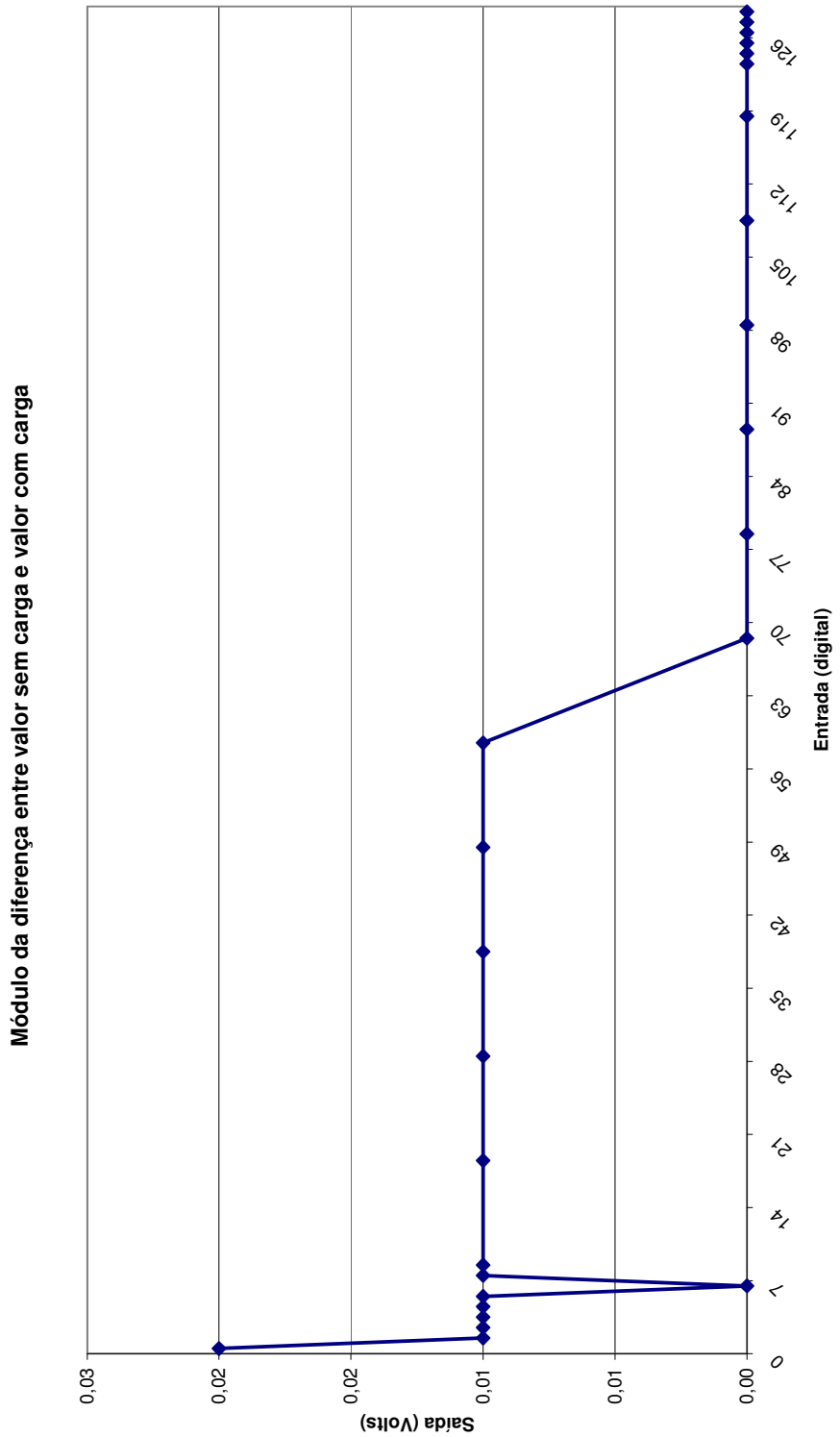


Gráfico 3.3-8 Diferença entre os valores sem carga e o valor com carga entre 0 e 128

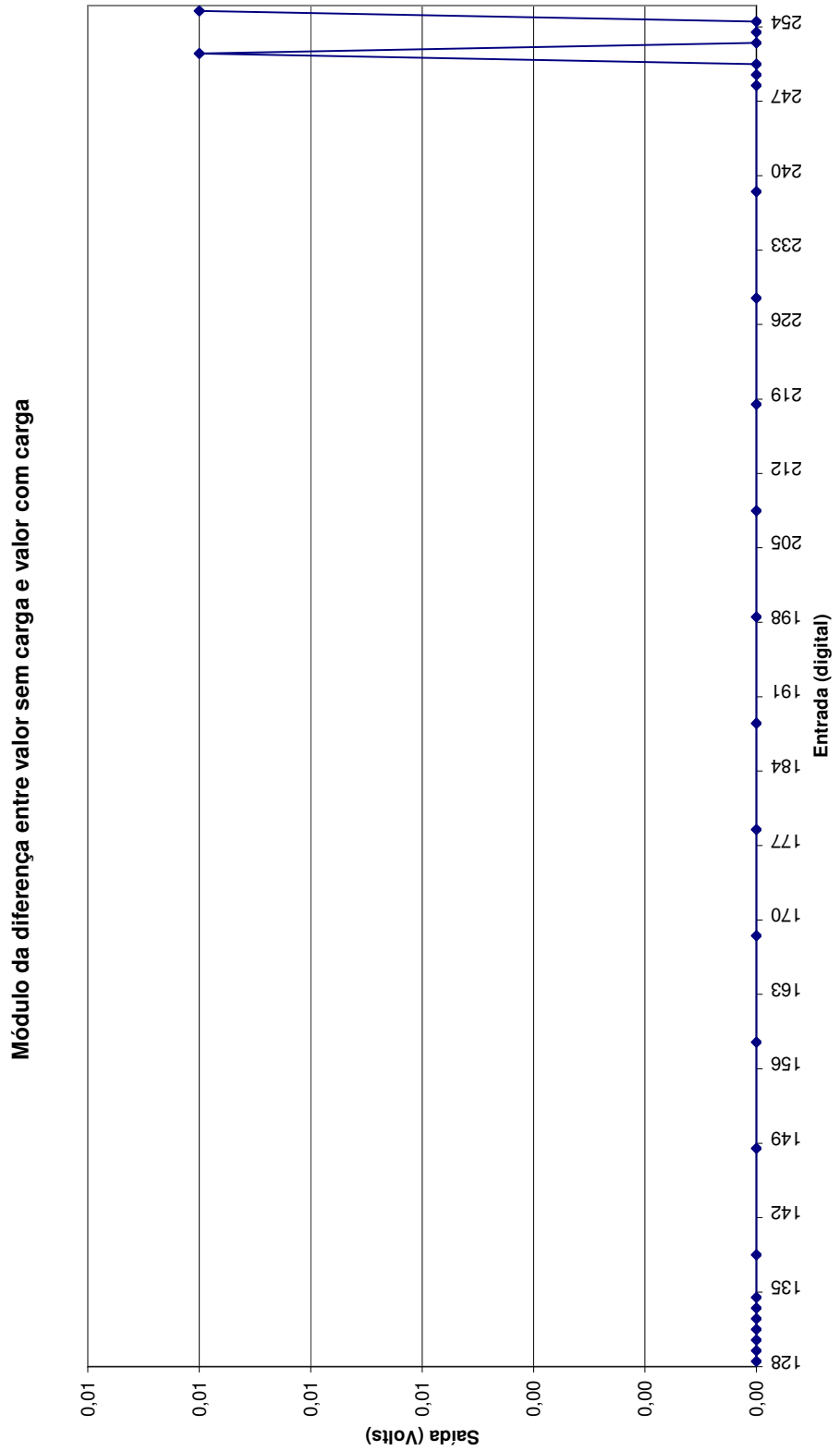


Gráfico 3.3-9 Diferença entre os valores sem carga e o valor com carga entre 128 e 255

3.4 Amplificador de potência (drive do motor)

3.4.1 Do motor DC

3.4.1.1 Esquema do amplificador usado

O amplificador foi projectado para ter um ganho de -1, (ganho = - R12/R9) tendo em conta que a principal função deste elemento é fornecer corrente ao circuito e não o de alterar o valor da tensão. A escolha por realimentação negativa teve como fundamento as vantagens desta configuração face à configuração não inversora, que são a estabilização do ganho face a variações de parâmetros devido à temperatura e redução de distorções e de efeitos de não linearidade.

A figura 3.4-1 representa o esquema eléctrico referente ao bloco do amplificador de potência.

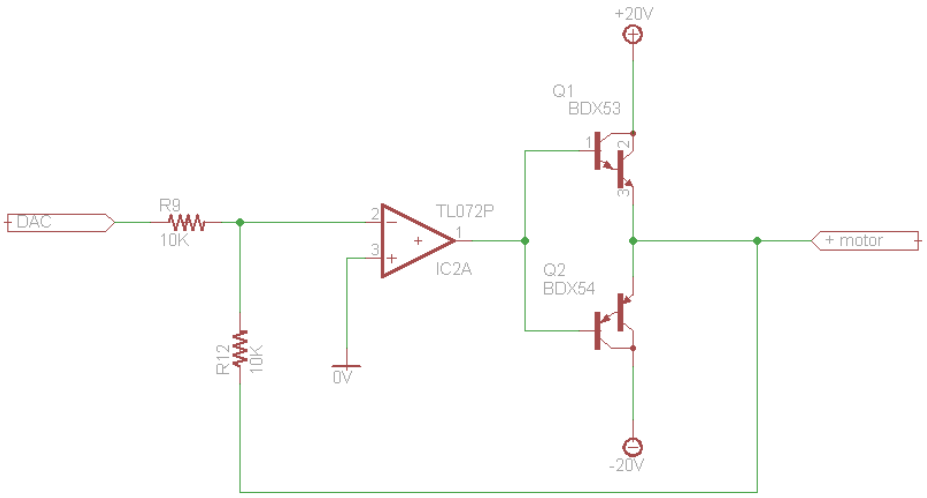


Figura 3.4-1 Esquema do amplificador de potência do motor DC

Mediram-se os valores de tensão na DAC e compararam-se com os valores de tensão que estava a ser entregue ao motor. Os valores obtidos estão na tabela 3.4-I os valores que estão na coluna referente à tensão na DAC são os medidos no ponto 3.3 que estão na tabela 3.3-II. Os valores que estão na coluna tensão do motor foram os valores medidos no motor para as diferentes tensões na DAC. A diferença entre os valores da coluna tensão na DAC e os valores da tensão no motor encontram-se nos gráficos 3.4-2 a 3.4-4.

Tabela 3.4-I Valores de tensão motor DC

Digital	Tensão na DAC			Tensão no motor DC
0-128	Teórico	Sem carga	Com motor	
0	-10,00	-10,00	-9,98	9,87
1	-9,92	-9,92	-9,91	9,80
8	-9,37	-9,38	-9,37	9,26
18	-8,59	-8,60	-8,59	8,49
28	-7,81	-7,82	-7,81	7,72
38	-7,02	-7,04	-7,03	6,95
48	-6,24	-6,26	-6,25	6,18
58	-5,46	-5,48	-5,47	5,41
68	-4,68	-4,67	-4,67	4,61
78	-3,89	-3,89	-3,89	3,84
88	-3,11	-3,11	-3,11	3,07
98	-2,33	-2,33	-2,33	2,30
108	-1,54	-1,55	-1,55	1,53
118	-0,76	-0,77	-0,77	0,75
124	-0,29	-0,31	-0,31	0,30
125	-0,21	-0,23	-0,23	0,21
126	-0,13	-0,15	-0,15	0,14
127	-0,06	-0,07	-0,07	0,06
128	0,02	0,02	0,02	0,01
129	0,10	0,10	0,09	0,08
130	0,18	0,17	0,17	0,16
131	0,26	0,25	0,25	0,24
132	0,34	0,33	0,33	0,32
133	0,41	0,41	0,41	0,39
134	0,49	0,49	0,48	0,47
148	1,59	1,58	1,58	1,54
158	2,37	2,35	2,35	2,30
168	3,15	3,14	3,14	3,08
178	3,94	3,92	3,92	3,85
188	4,72	4,70	4,70	4,62
198	5,50	5,51	5,51	5,42
208	6,29	6,29	6,29	6,19
218	7,07	7,07	7,07	6,96

228	7,85	7,85	7,85	7,73
238	8,64	8,63	8,63	8,50
248	9,42	9,41	9,41	9,27
254	9,89	9,88	9,88	9,73
255	9,97	9,96	9,95	9,81

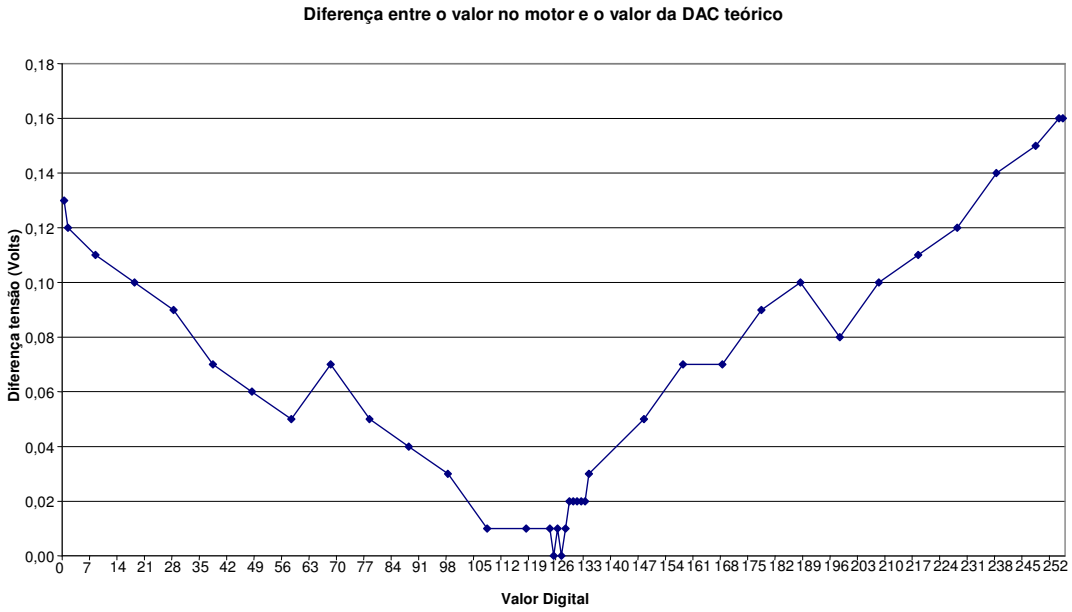


Figura 3.4-2 Diferença entre o valor no motor e o valor da DAC teórico

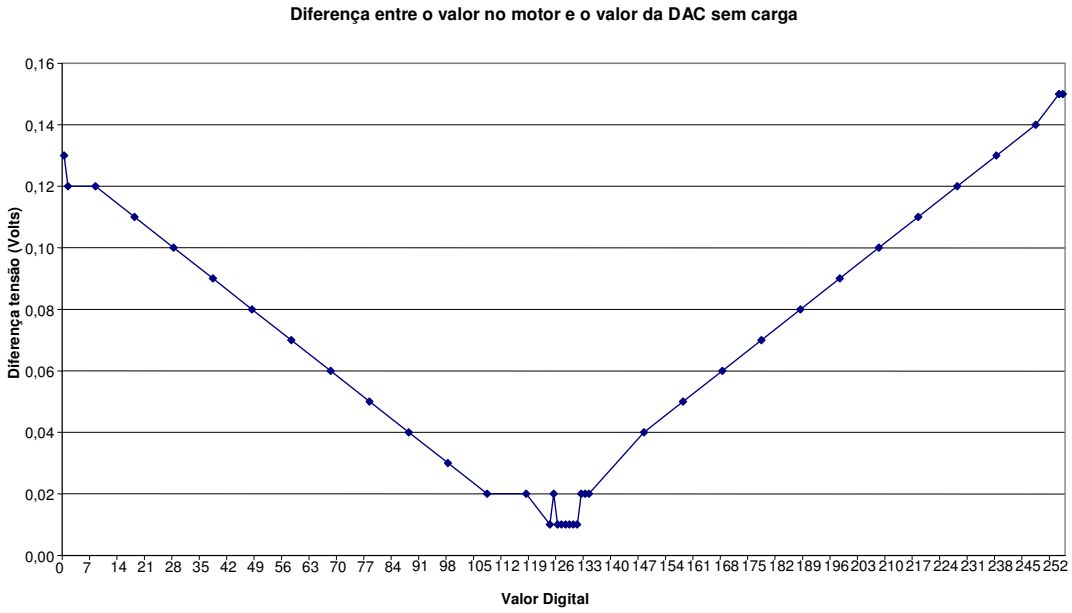


Figura 3.4-3 Diferença entre o valor no motor e o valor da DAC sem carga

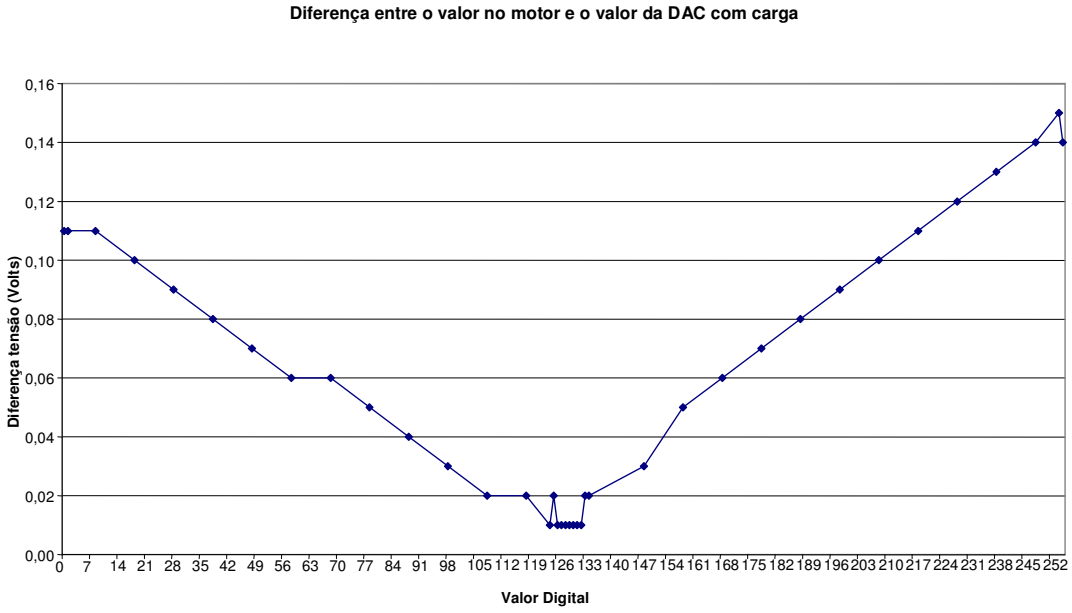


Figura 3.4-4 Diferença entre o valor no motor e o valor da DAC com carga

Por análise dos gráficos podemos verificar que quanto maior for a tensão aplicada ao motor maior é a diferença entre o valor pretendido e o valor medido.

3.4.2 Do motor passo-a-passo

Para fazer o drive de controlo e potência deste motor usaram-se os circuitos integrados L297 e o L298N. O esquema usado encontra-se na figura 3.4-2.

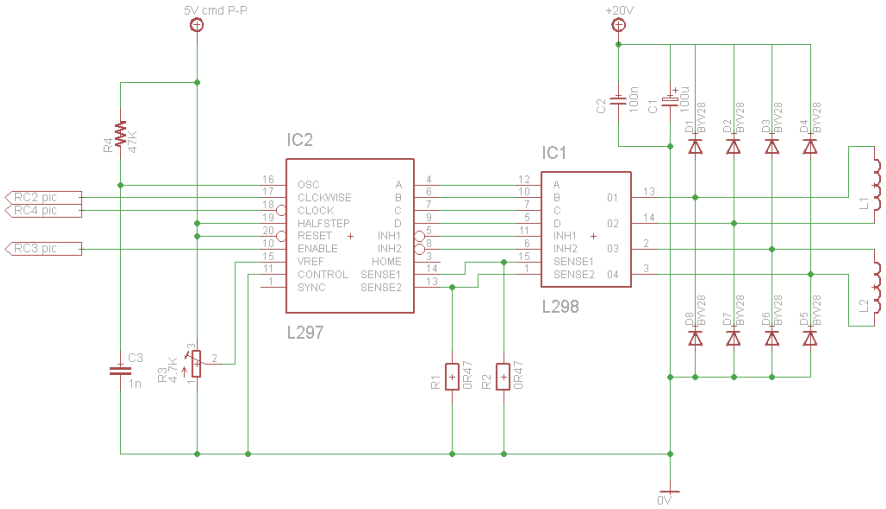


Figura 3.4-5 Esquema do drive do motor passo-a-passo

O motor passo-a-passo irá funcionar em meio passo, para ter um maior controlo de deslocamento. Este circuito permite fornecer uma corrente máxima de 2A em modo contínuo.

3.5 Sensores de velocidade e de binário

3.5.1 Sensor de velocidade

3.5.1.1 Para o motor DC

Para o motor DC o sensor de velocidade escolhido foi o *encoder*. Este foi escolhido porque o motor que se escolheu tinha acoplado um *encoder* sendo que a tracção entre o veio do motor e o *encoder* é muito boa. Inicialmente utilizou-se um motor e um *encoder* que estavam acoplados por um veio exterior e verificou-se que se o meio de acoplamento entre os veios não for boa vai haver oscilações nas medições da velocidade que torna o sistema pouco viável.

No projecto o *encoder* utilizado é um *encoder* incremental com dois canais. Como a saída resultante do *encoder* usado é uma onda quadrada que varia entre 5 e 0 Volts, estes sinais podem ser ligados directamente à PIC.

Para calcular a velocidade de rotação do motor aproveitou-se apenas um canal ligado ao pino RC0 que corresponde ao T1CK1, que permite através de interrupções sempre que há uma transição de 0 para 1 contar o número de impulsos. A forma de converter estes impulsos em velocidade está descrita mais à frente no capítulo de software da PIC.

Para determinar o sentido de rotação inicialmente ligou-se o outro canal a um pino da PIC, mas esta solução apresentou problemas que se traduziam em bloqueio do programa quando a velocidade era muito baixa. Para resolver esse problema montou-se um circuito que converte o sentido de rotação num valor digital, basicamente é um circuito de *set* e *reset*. O esquema do circuito que dá o sentido de rotação encontra-se da figura 3.5-1.

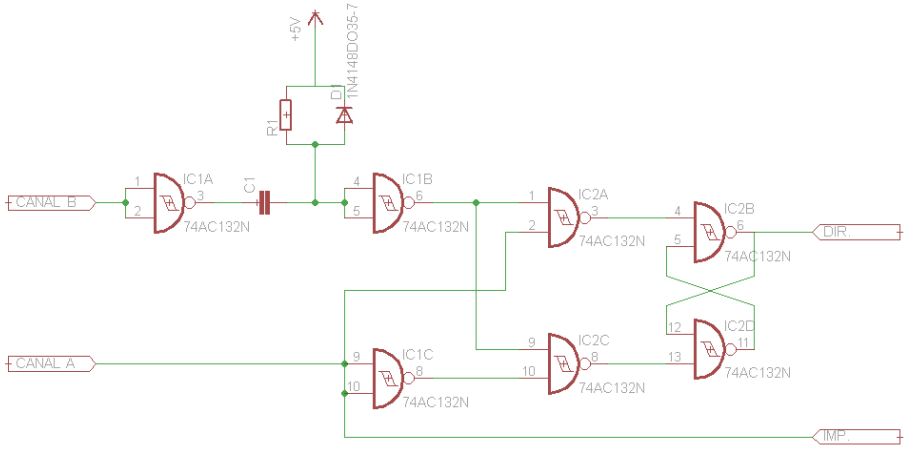


Figura 3.5-1 Circuito eléctrico do sentido de rotação

3.5.1.2 Para o motor passo-a-passo

Para o motor passo-a-passo o método escolhido para medir a velocidade é semelhante à do motor DC, mas em vez de utilizar um *encoder* utilizou-se um outro motor passo-a-passo a servir de sensor de velocidade.

Quando rodamos um motor passo-a-passo este gera tensão mas suas bobinas e esta tensão é parecida com a onda quadrada que se obtém no *encoder*. A principal diferença é que enquanto num *encoder* a tensão da onda é fixa, num motor passo-a-passo quanto maior for a velocidade maior é a tensão obtida pelas bobines podendo estas chegarem a valores das centenas de volts facilmente.

O circuito usado para converter a saída do motor passo-a-passo numa onda quadrada com tensão entre os 0 e os 5 volts foi o da figura 3.5-2. Este circuito tem com base uma montagem do OPAMP no modo saturado. O valor das resistências R1e R8 tem que ser elevado porque o valor da tensão gerado pelas bobines do motor atingem tensões elevadas, caso contrário estas queimam-se devido à grande queda de tensão que resulta em potência dissipada.

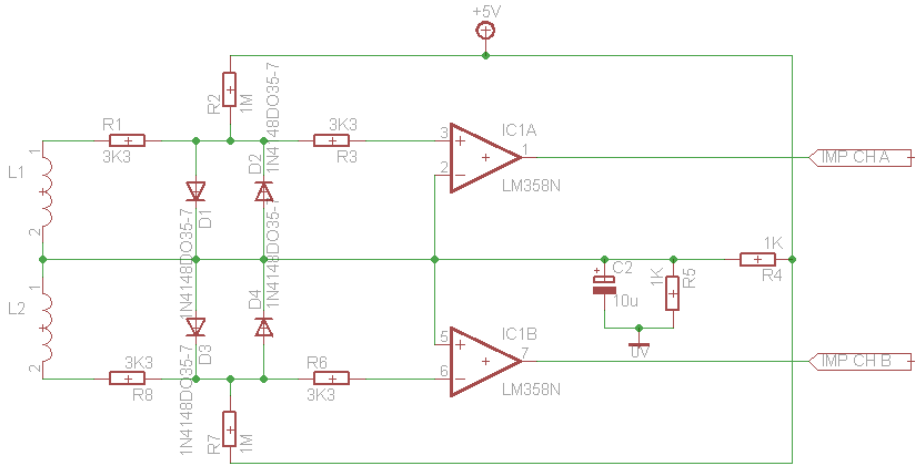


Figura 3.5-2 Gerador de onda quadrada entre 0 e 5 Volts

Os díodos no esquema da figura 3.5-2 servem para garantir que a tensão de entrada nos OPAMP's não é superior a 0,7V porque caso contrário poderia levar a destruição dos OPAMP's. Com este circuito temos dois canais desfasados de 90º entre si por análise da tabela da Figura 3.5-3 vemos que no primeiro passo a bobine 1 fica ligada no passo seguinte as duas bobines ficam ligadas em seguida a bobina 1 desliga-se e a bobina 2 continua ligada e assim sucessivamente, assim temos duas ondas quadradas desfasadas entre si de 90º.

Como já não temos disponíveis na PIC entradas que permitam fazer a contagem de impulsos através do timer 1 optou-se por usar um circuito integrado que converte estes impulsos em frequência e esta frequência é depois convertida numa tensão analógica que varia entre 0 e 5 volts. Como sabemos que a frequência é directamente proporcional à velocidade, e como temos ainda algumas saídas analógica para digital livres na PIC optou-se por esta solução para medir a velocidade de rotação do motor passo-a-passo. O esquema eléctrico deste circuito encontra-se na figura 3.5-2.

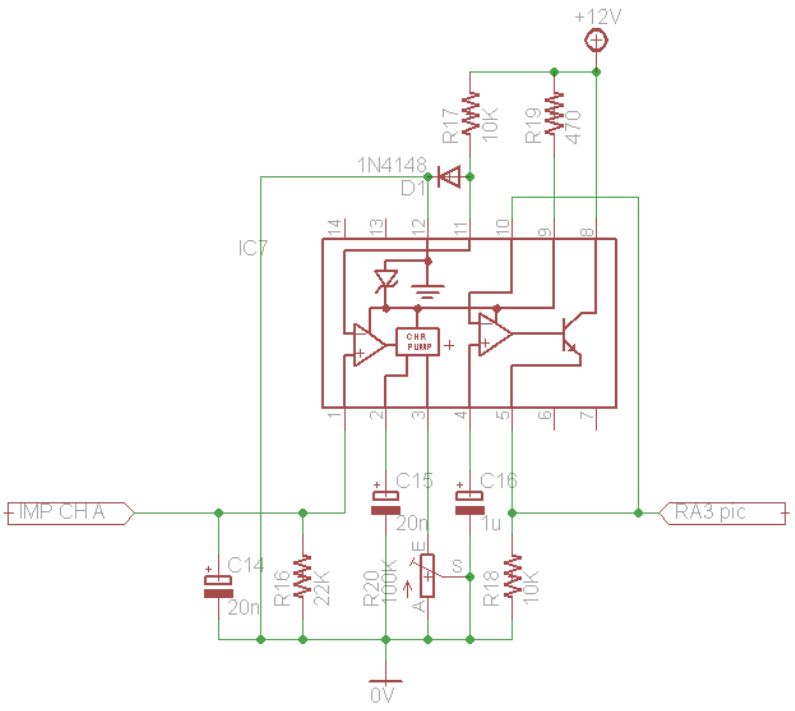


Figura 3.5-3 Esquema do conversor de impulsos (frequência) em tensão

3.5.2 Sensor de binário

Para se medir o binário de uma forma correcta teríamos que recorrer a um transdutor de binário pelo que não foi possível em tempo útil abordar este assunto. Sabendo que existe uma relação teórica para a obtenção do binário, é possível obter o binário através da velocidade, corrente e a tensão aplicada ao motor, construiu-se elementos capazes de medir velocidade, corrente e a tensão aplicada ao motor.

3.5.2.1 Medidor de corrente

A maneira mais fácil de medir a corrente aplicada ao motor é de medir a queda de tensão numa resistência de baixo valor com valor conhecido, colocada entre o motor e a tensão de referência (0V). Desta forma a corrente que é aplicada ao motor é obtida imediatamente pela lei de ohm, I=V/R. Quanto menor o valor da resistência menor será a queda de tensão provocada e menor será a interferência com a tensão aplicada ao motor.

Para que se possa aplicar a lei e converter a queda de tensão nas resistências em corrente é necessário primeiro converter essa tensão num valor analógico compreendido entre 0 e 5 volts. Isto porque a PIC tem portas que convertem tensão analógica em digital, por isso utilizou-se essa vantagem da PIC.

Pretende-se medir correntes que variem entre -4A e 4A, para isso vamos fazer corresponder -5V a -4A 0V a 0A e 5V a 4A. Optou-se por escolher uma resolução de 10 bits na conversão de analógico para digital para ter uma maior resolução.

A resolução que se consegue obter é de, $4000-(-4000)/1024=7{,}8mA$, conseguimos por isso medir correntes múltiplas de 7,8mA desde -4A até 4A. Para uma resistência de 1Ω o ganho que o amplificador terá que ter é de $5/(1*4)=1{,}25V/V$, a queda de tensão na resistência é dada pelo valor da resistência vezes a corrente que a atravessa.

Nas tabelas seguintes está representado os vários valores que se obtêm e que são necessários para projectar o medidor de corrente.

Tabela 3.5-I Tabela de conversão binário -> valor obtido na PIC

Binário	Decimal	Tensão [V]	Corrente ±4000[mA]	Corrente ±2000[mA]	Conversão [V]
0000000000	0	-5	-4000	-2000	0
1000000000	512	0	0	0	2,5
1111111111	1023	5	4000	2000	5

Na tabela 3.5-I está representado para alem da correspondência da valor binário para decimal o valor da tensão amplificado que se obtém para os respectivos valores de corrente, os valores de corrente que se obtém para uma escala que varie entre -4 e 4A e para uma escala que varie entre -2 e 2A, assim como o respectivo valor convertido num valor que possa ser convertido pela PIC, que conforme já referido tem que variar entre 0 e 5V.

O valor do ganho que o amplificador tem que ter é dado por Valor máximo de saída/(queda de tensão máxima), para o caso de querermos o máximo de 4ª, temos ganho=5/(1x4)=1,25V/V. O valor amplificado é dado pelo valor da queda de tensão na resistência vezes o ganho. O valor convertido é o valor amplificado a multiplicar por 0,5+2,5. O valor correspondente da conversão de tensão para analógico é dado por convertido x(1023/5). Os 1023 são o valor máximo que se obtém com 10 bits.

Na tabela 3.5-II encontram-se alguns valores de corrente e o seu correspondente valor digital, assim como os cálculos intermédios.

Tabela 3.5-II Tabela de valores para uma corrente de +/- 4A

Escala máxima 4ª						
Corrente [mA]	Resistência [Ohm]	Ganho	ΔV resistência [V]	Amplificado [V]	Convertido [V]	DEC [PIC] 10BIT
-4000	1	1,25	-4	-5	0	0
-2000	1	1,25	-2	-2,5	1,25	256
0	1	1,25	0	0	2,5	512
2000	1	1,25	2	2,5	3,75	767
4000	1	1,25	4	5	5	1023

Na tabela 3.5-III encontram-se alguns valores de corrente e o seu correspondente valor digital, assim como os cálculos intermédios, para uma corrente a variar entre -2A e 2A.

Tabela 3.5-III Tabela de valores para uma corrente de +/- 2A

Escala máxima 2 A						
Corrente [mA]	Resistência [Ohm]	Ganho	ΔV resistência [V]	Amplificado [V]	Convertido [V]	DEC [PIC] 10BIT
-2000	1	2,5	-2	-5	0	0
-1000	1	2,5	-1	-2,5	1,25	256
0	1	2,5	0	0	2,5	512
1000	1	2,5	1	2,5	3,75	768
2000	1	2,5	2	5	5	1023

A figura 3.5-4 representa o esquema eléctrico montado para o medidor de corrente.

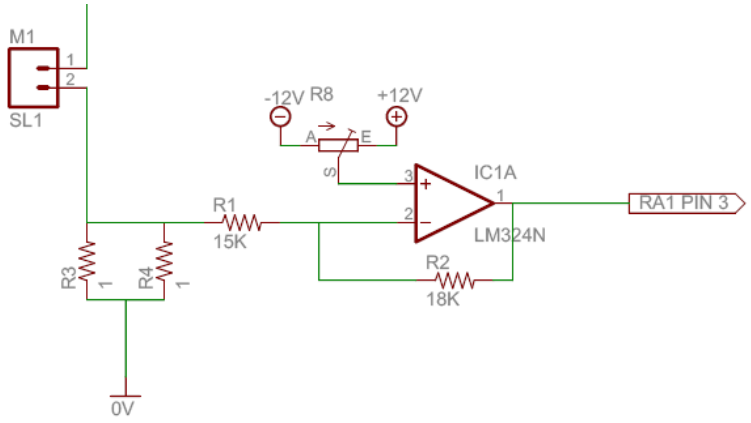


Figura 3.5-4 Esquema eléctrico do medidor de corrente

No pino 3 do OPAMP está a tensão de *offset* que serve para ajustar a minha tensão que irá corresponder a zero amperes. O ganho é dado por R2/R1 $\approx$1,2.

3.5.2.2 Medidor de tensão aplicada ao motor

O princípio base deste medidor é equivalente ao do ponto 3.5.2.1, mas em vez de medir corrente queremos medir tensão. Para medirmos a tensão vamos verificar a tensão que é aplicada no motor em relação ao ponto de referência, zero volts. O valor do ganho necessário é dado por 5/12, em que 5 volts que correspondem ao valor máximo de saída e 12 volts o valor máximo da entrada. A resolução deste medidor é de 23,44mV. Em tudo o resto é semelhante ao calculado no ponto 3.5.2.1. Nas tabelas seguintes está ilustrado os valores calculados.

Tabela 3.5-IV Tabela de conversão binário -> valor medido na PIC

Binário	Decimal	Tensão [V]	Tensão ±12V	Conversão [V]
0000000000	0	-5	-12	0
1000000000	512	0	0	2,5
1111111111	1023	5	12	5

Na tabela 3.5-IV o valor correspondente a coluna Tensão[V] diz respeito a tensão depois de amplificada. A tabela 3.5-V representa os diversos valores para uma tensão de ±12V convertidos para valor digital.

Tabela 3.5-V Tabela de valores para uma tensão de +/- 12V

Escala máxima 12V				
Corrente [mA]	Ganho	Amplificado [V]	Convertido [V]	DEC [PIC] 10BIT
-12	0,416666667	-5	0	0
-6	0,416666667	-2,5	1,25	256
0	0,416666667	0	2,5	512
6	0,416666667	2,5	3,75	767
12	0,416666667	5	5	1023

O esquema eléctrico deste medidor encontra-se na figura 3.5-5.

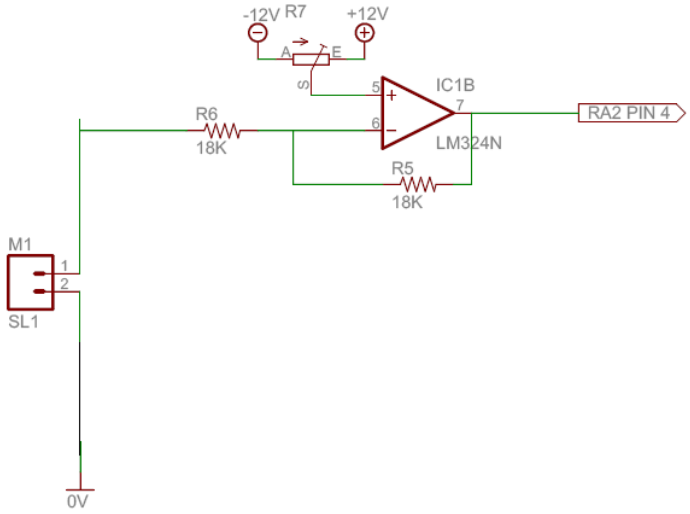


Figura 3.5-5Esquema eléctrico do medidor de tensão

3.6 Unidade de processamento

3.6.1 PIC 16F877

O microcontrolador da Microchip PIC 16F877 foi o escolhido para realizar este trabalho. Este microcontrolador foi usado com um velocidade de relógio de 20 MHz, e a comunicação escolhida com o computador externo foi a USART (*Universal Synchronous /Asynchonous Receiver Transmitter*), com uma taxa de transmissão, *baud rate*, de 19200 bits/segundo.

O microcontrolador suporta uma velocidade de relógio de 20 MHz, possui interrupções de hardware, 5 portos, 3 *timers*, 8 entradas conversoras de analógico para digital (ADC), comunicação USART, 2 saídas PWM / Captura / Comparação.

O código usado para programar a PIC foi o C, e o programa para escrever e compilar foi o MPLAB da Microchip. Usar código C possibilita um alto nível de interface entre o software e o hardware.

3.6.1.1 O timer 0

O módulo timer 0 tem as seguintes características:

- 8 bits temporizador/contador
- De escrita e leitura
- Pré-divisor programado por software
- Interrupção por excesso de 0xFF a 0x00

O timer 0 tem um registo associado de 8 bits chamado TMR0, onde se pode escrever o valor que queremos que ocorra a interrupção por excesso, o valor máximo que se pode atribuir a este registo é 255.

O registo TMR0IF e a flag de interrupção associadas ao timer 0, quando o TMR0 ultrapassa o valor limite.

O timer 0 tem um registo associado, o OPTION, que permite seleccionar o modo de funcionamento do timer. Este registo encontra-se na figura 3.6-1.

OPTION_REG REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Figura 3.6-1 Registo do timer 0

Os registos PS0:2 permitem definir o pré-divisor pretendido, a tabela com o valor associado aos registo qual a divisão que se obtêm encontra-se na figura 3.6.2. O PSA definir qual o relógio a que se faz a divisão. O registo T0SE define se o contador incrementa no flanco descendente ou ascendente. T0CS define se o TMR0 incrementa pelo relógio interno ou externo.

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Figura 3.6-2 Registos PS0:2

A frequência em que se vão gerar interrupções é dada por $f_{out} = \frac{f_{clk}}{4 \times divisao \times (256 - TMR0)}$, em que f_{clk} é a frequência do relógio seleccionado.

3.6.1.2 O timer 1

O módulo timer 1 da PIC, pode funcionar como temporizador ou contador e tem as seguintes características:

- 16-bit temporizador/contador que usam os dois registos de 8-bit cada, o TMR1H e o TMR1L.

- Permite leitura e escrita
- Pré-divisão do relógio programável por software
- Pode usar o relógio interno ou um relógio externo
- Permite interrupções

O timer 1 tem um registo chamado TMR1 com tamanho de 16 bits. O TMR1 consiste em dois registos de 8 bits cada um, o TMR1H e o TMR1L, por isso o TMR1 pode ir de 0x0000 a 0xFFFF (de 0 a 65535 decimal). Se as interrupções do timer 1 estiverem activas e ocorrer um valor do TMR1 superior a 0xFFFF vai activar a flag TMR1IF (PIR1<0>) e activar uma interrupção. As interrupções podem ser desactivadas ou activadas através do registo PIE1 do TMR1IE.

Pode-se escolher usar o relógio interno da PIC ou um relógio externo ligado ao pino RC0 da PIC. O registo T1CON que se encontra na figura 3.6-3, é o registo no qual se vai definir o modo de funcionamento do timer1.

T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

Figura 3.6-3 Registo do timer 1

O registo TMR1ON é o bit que permite activar ou desactivar o timer. Se TMR1ON estiver a um activa o timer, se estiver a zero o timer fica desligado.

O registo TMR1CS é o registo que nos permite seleccionar o modo de funcionamento do timer, se este valor estiver a um o timer vai contar os impulsos que aparecem no pino RC0 a cada flanco ascendente. Se o registo estiver a zero contador de impulsos vai ser o relogio interno da PIC.

O registo T1SYNC é usado para seleccionarmos se pretendemos que haja sincronização com o relógio externo, se o bit estiver a um não há sincronia caso contrário há sincronia.

O registo T1OSCEN bit que nos permite seleccionar se o oscilador activo ou desactivo.

Os registos T1CKPS1:T1CKPS0 são os bits que permitem definir a divisão relógio interno da PIC, o valor que os registos podem ter e qual a divisão associada são:

00 = divisão por 1

01 = divisão por 2

10 = divisão por 4

11 = divisão por 8

3.6.1.3 O timer 2

O módulo timer 0 tem as seguintes características:

- 8 bits temporizador/contador
- De escrita e leitura
- Pré-divisor e pós-divisor programado por software
- Interrupção por excesso de 0xFF a 0x00
- Relógio interno apenas

Este registo encontra-se na figura 3.6-4.

T2CON: TIMER2 CONTROL REGISTER (ADDRESS 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

Figura 3.6-4 Registo do timer 2

Os registos T2CKPS0:1 permitem definir o pré-divisor pretendido, a tabela com o valor associado aos registo qual a divisão que se obtêm encontra-se na figura 3.6-5.

```
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16
```

Figura 3.6-5 Registos T2CKPS0:1

O TMR2ON permiti activar ou desactivar o timer 2. Os registos TOUTPS0:3 definem o pós divisor associado, a tabela com o valor dos registos TOUTPS0:3 e a que pós-divisor representam encontram-se na figura 3.6-6.

```
0000 = 1:1 Postscale
0001 = 1:2 Postscale
0010 = 1:3 Postscale
•
•
•
1111 = 1:16 Postscale
```

Figura 3.6-6 Registos TOUTPS0:3

A frequência em que se vão gerar interrupções é dada por $f_{out} = \frac{f_{clk}}{4 \times pré\_divisão \times (PR2 - TMR2) \times pós\_divisão}$ em que f_{clk} é a frequência do relógio

seleccionado. Os registos associados ao timer 2 que indicam interrupções são o TMR2IF e o TMR2IE.

3.6.2 MATLAB

MATLAB é um software interactivo de alta performance voltado para o cálculo numérico. O MATLAB integra análise numérica, cálculo com matrizes, processamento de sinais, comunicação série e construção de gráficos em ambiente fácil de usar onde problemas e soluções são expressos somente como eles são escritos matematicamente, ao contrário da programação tradicional.

O GUI (interface gráfico com o usuário) do MATLAB foi usado para permitir um controlo simples sobre o sistema a controlar. Tanto no controlo do motor DC como do motor passo-a-passo é necessário primeiro abrir com sucesso a comunicação entre o PC e a PIC. Para isso é necessário seleccionar a porta série correspondente em opções e em seguida pressionar em teste com. Em seguida aparece uma janela com o resultado do teste, se obteve comunicação aparece a seguinte mensagem “Ligação estabelecida”, caso contrario aparece uma mensagem a dizer “Não é possível abrir a porta”. Para fechar a porta de comunicação pressionar em “close com”.

Na figura 3.6-7 podemos ver o ambiente gráfico para o controlo do motor DC, na medição em malha aberta.

Figura 3.6-7 Ambiente gráfico do MATLAB motor DC

Com este ambiente podemos em opções modificar o tempo de amostragem dos sinais a serem lidos pela PIC, e definir a porta série que está a ser usada para comunicação. Em definições do sinal podemos escolher qual o tipo de sinal que pretendemos aplicar ao motor. Os tipos se sinais disponíveis são: impulso, degrau, rampa e seno. A amplitude refere-se à tensão máxima a aplicar ao motor. A frequência quando se aplica refere-se ao intervalo dos sinais. O tempo de *Run* é o tempo que o sinal será aplicado ao motor. Quando o motor está em teste, estado de RUN, os valores instantâneos da velocidade tensão e corrente do motor aparece na janela de multímetros, em simultâneo um gráfico com os valores da velocidade e tensão aplicadas ao motor é desenhado no gráfico que aparece na figura 3.6-7.

Para o controlo do motor passo-a-passo foi criado um outro ambiente gráfico que se encontra na figura 3.6-8. Com este ambiente pode-se indicar o sentido de rotação e o intervalo de impulsos que se pretende atingir. Quando se pressiona em RUN irá visualizar-se a velocidade de rotação do motor durante o tempo que está em tempo de Run, o valor da velocidade instantânea aparecerá no campo Velocidade em multímetro, e um gráfico representado a evolução do velocidade no tempo na área de gráfico do ambiente gráfico.

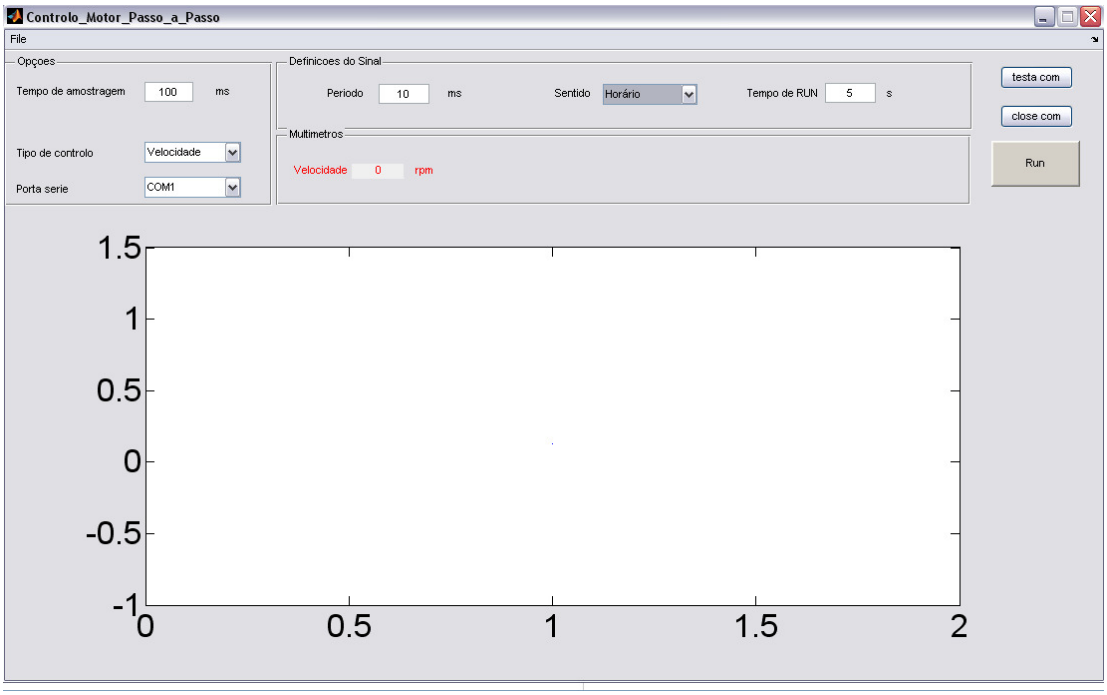


Figura 3.6-8 Ambiente gráfico do MATLAB motor Passo-a-passo

3.7 O controlador PID

No caso de o sistema projectado não cumprir as especificações desejadas e numa primeira tentativa de obter um sistema operacional, procuramos estabelecer um compromisso entre especificações, muitas delas conflituosas, através do ajuste dos parâmetros do sistema obtido de modo a ver se conseguimos um desempenho aceitável.

Verifica-se contudo que, muitas vezes não é possível só através do ajuste dos parâmetros do sistema construído, atingir o objectivo pretendido. Torna-se necessário ir mais além e reconsiderar a estrutura do sistema procedendo à alteração do seu projecto de modo a que se obtenha o desempenho desejado. Isto é torna-se necessário reexaminar e alterar o projecto inicial introduzindo novas componentes ou redesenhando as existentes.

O conjunto de operações anteriores, ajuste dos parâmetros ou alteração da estrutura de um sistema de controlo de modo a se garantir o seu desempenho de uma forma apropriada, designa-se por compensação do sistema.

Os parâmetros que normalmente são ajustados tendo em vista compensar um sistema são: o ganho do controlador ou o da unidade de realimentação. As especificações do desempenho pretendido para um sistema de controlo podem ser expressas quer em termos de especificações no domínio dos tempos quer no domínio da frequência ou ainda em termos estatísticos (controlo estocástico).

No domínio dos tempos podemos, por exemplo para a componente transitória da resposta correspondente a uma entrada em degrau, limitar o tempo de pico, a percentagem de sobre-elevação e/ou tempo de estabelecimento a x%. Ainda no domínio dos tempos é também usual especificar o erro em regime estacionário máximo admissível para vários sinais padrão de entrada e/ou de perturbação.

O ganho variável Kp que vamos introduzir corresponderá a um compensador que se designa por compensador proporcional (compensador P) uma vez que o sinal que produz na sua saída é proporcional ao sinal que lhe é aplicado.

O ganho variável Ki que vamos introduzir corresponderá a um compensador que se designa por compensador integrador (compensador I) uma vez que o sinal que produz na sua saída corresponde à integração (acumulação ao longo do tempo) do sinal que lhe é aplicado.

O ganho variável Kd que vamos introduzir corresponderá a um compensador que se designa por compensador derivativo (compensador D) uma vez que o sinal que produz na sua saída corresponde à derivação do sinal que lhe é aplicado.

De forma a manter a velocidade de rotação do motor DC constante, ou pelo menos dentro de um intervalo considerado constante, pode-se recorrer a implementação do controlador PID. O comportamento que cada constante irá ter no sistema encontra-se na figura 4.2-8. [10]

Resposta ao degrau	**Tempo de crescimento**	**Sobre-elevação**	**Tempo de estabeleci-mento**	**Erro em regime estacionário**
Aumentando **Kp**	diminui	aumenta	pequena variação	diminui
Aumentando **Ki**	diminui	aumenta	aumenta	elimina
Aumentando **Kd**	pequena variação	diminui	diminui	pequena variação

Figura 3.7-1 Efeito da variação dos parâmetros do compensador PID

3.7.1 Adaptação do Controlador PID analógico

A dedução da fórmula do controlador PID (Proporcional, Integrativo e Derivativo) digital, através da formula clássica controlador PID analógico, e aqui abordada. Desta forma, é possível utilizar os métodos convencionais de determinação dos parâmetros do controlador analógico, e depois adapta-los ao PID digital.

Obviamente, isto só resulta se o período de amostragem / actuação for suficientemente fino em relação às constantes de tempo do sistema e do controlador analógico original. Se esta condição não se verificar é necessário utilizar outras técnicas para determinar as constantes do PID digital. No entanto o código de implementação do PID digital é obviamente o mesmo.
A partir da equação do PID:

$$c(t) = Kp \times e(t) + Ki \int_0^t e(t)dt + Kd \frac{de(t)}{dt}$$

onde c(t) é o sinal de entrada, e(t) o erro entre a saída e o valor esperado na saída e KP, KI e KD as constantes proporcional, integral e derivativa do PID, a equação do PID digital é obtida substituindo a variável tempo (t) por:

$$t = nT$$

onde T é o período de amostragem e n o numero da amostra. Obtêm-se assim a equação:

$$c(n) = Kp \times e(n) + Ki \times T \times \sum_{j=0}^{n} e(jT) + Kd \times \frac{e(nT) - e((n-1)T)}{T}$$

Como nT representa a amostra número do sistema, é substituída simplesmente por n, dando origem á primeira equação do PID digital:

$$c(n) = Kp \times e(n) + Ki \times T \times \sum_{j=0}^{n} e(jT) + Kd \times \frac{e(n) - e(n-1)}{T}$$

4 Software desenvolvido

4.1 Software da PIC

4.1.1 Envio e recepção de mensagens entre a PIC e o PC

A estrutura das mensagens a serem enviadas entre a PIC e o PC tem o seguinte apresentado na tabela 4.1-I.

Tabela 4.1-I Estrutura das mensagens

Formato	Inicio	Tipo	Comprimento do conteúdo	Conteúdo	Fim
Hexadecimal	0xAA	0xXX	0x??	????	0x55

Todas as mensagem começam pelo valor hexadecimal 0xAA, em seguida vem o tipo de mensagem, as mensagem que são enviadas da PIC para o PC tomam o valor de 0x0X, em que X representa o tipo de mensagem, os respectivos valores de X para as diferentes mensagens encontra-se na tabela 4.1-II. Os conteúdos dos campos inicio, tipo, comprimento do conteúdo e fim de mensagem ocupam todos 8 bits, 1 byte, o número de bytes do campo conteúdo é o único que é variável, dependendo do tamanho que do valor do conteúdo tenha, mas o tamanho é sempre múltiplo do byte. No caso de uma mensagem que peça apenas informação, o conteúdo não tem importância, basta dizer o tipo de mensagem para que o receptor saiba o que deve fazer.

Tabela 4.1-II Tipos de mensagens e valores atribuídos

PIC-> PC			PC-> PIC		
Tipo	Valor (hex)	Valor (dec)	Tipo	Valor (hex)	Valor (dec)
Retorno do teste de comunicação	0x00	0	Teste de comunicação	0x10	16
Retorno tempo de amostragem	0x01	1	Tempo de amostragem	0x11	17
Não está definida	0x02	2	Liga/desliga motor DC	0x12	18

Não está definida	0x03	3	Envio de tensão para o motor DC	0x13	19
Envia o valor da velocidade do motor DC	0x04	4	Pede a leitura da velocidade do motor DC	0x14	20
Envia o valor da corrente do motor DC	0x05	5	Pede a leitura da corrente do motor DC	0x15	21
Envia o valor da tensão do motor DC	0x06	6	Pede a leitura da tensão do motor DC	0x16	22
Não está definida	0x07	7	Activar o PID da PIC	0x17	23
Não está definida	0x08	8	Liga/desliga motor passo-a-passo	0x18	24
Não está definida	0x09	9	Envio de tensão para o motor passo-a-passo	0x19	25
Não está definida	0x0A	10	DÁ O SENTIDO DE ROTAÇAO	0x1A	26
Envia o valor da velocidade do motor passo-a-passo	0x0B	11	Pede a leitura da velocidade do motor passo-a-passo	0x1B	27
Não está definida	0x0C	12	Tempo de controlo PID	0X1C	28
Não está definida	0x0D	13	Kp	0X1D	29
Não está definida	0x0E	14	Ki	0X1E	30
Não está definida	0x0F	15	Kd	0X1F	31

4.1.2 Implementação dos programas da PIC

4.1.2.1 Recepção de mensagens na PIC

O fluxograma do modo de captura da mensagem encontra-se na figura 4.1-1. Sempre que existe um carácter a ser recebido pela PIC este gera uma interrupção e lê o carácter recebido para uma string que vai guardar a mensagem. Quando tem uma mensagem completa, activa uma flag que vai indicar que já existe uma mensagem pronta a ser processada. A rotina RSI é a rotina de atendimento as interrupções.

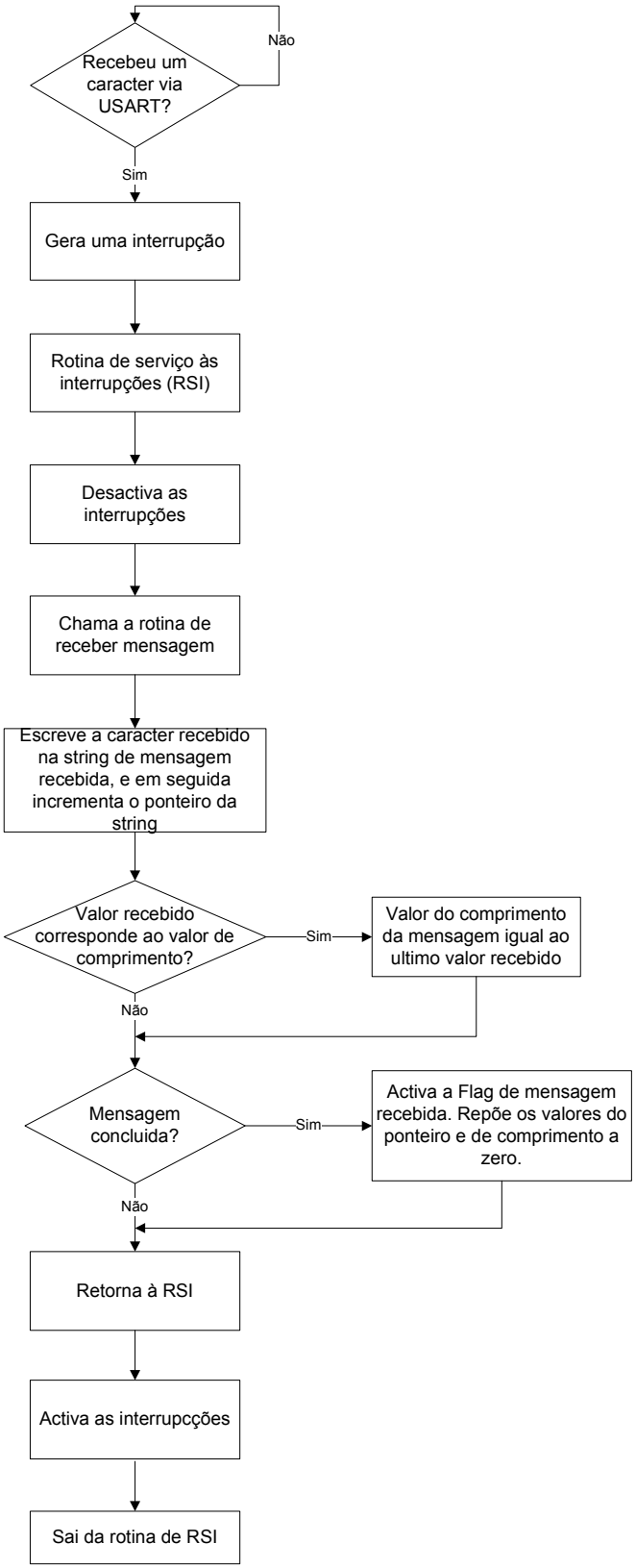


Figura 4.1-1Fluxograma da leitura de mensagem na PIC

O fluxograma do modo de processamento das mensagens recebidas encontra-se na figura 4.1-2, este ciclo é despoletado na função main, que está sempre a verificar o estado da flag de mensagem recebida.

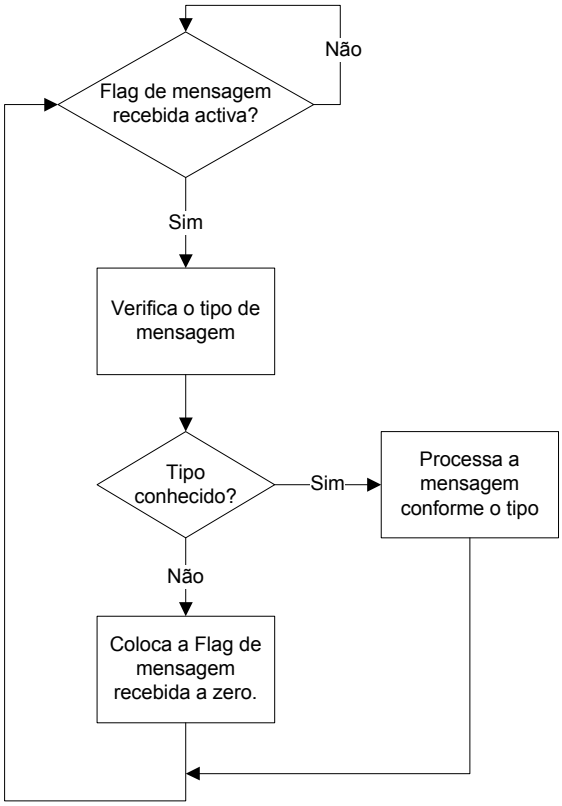


Figura 4.1-2 Fluxograma processamento das mensagens

4.1.2.2 Retorno do teste de comunicação

O fluxograma do programa que verifica as comunicações encontra-se na figura4.1-3. Esta função serve para verificar se as comunicações entre a PIC e o PC estão a funcionar como previsto.

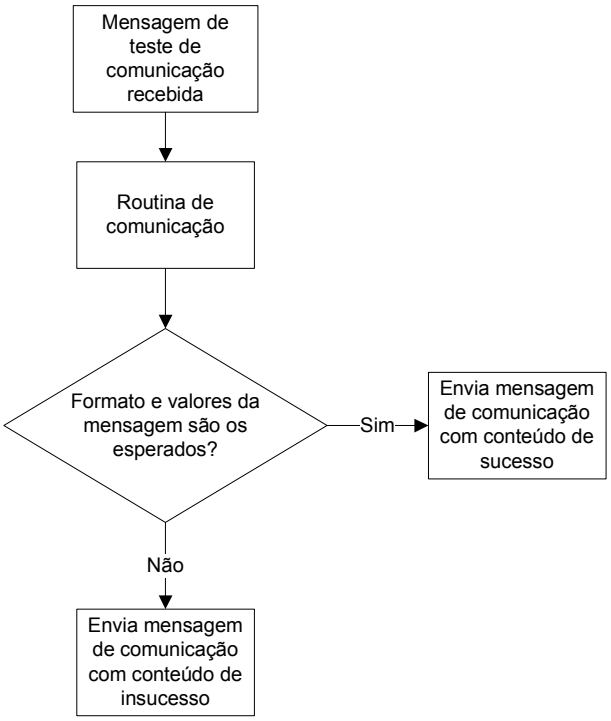


Figura 4.1-3 Fluxograma do retorno do teste de comunicação

4.1.2.3 Retorno tempo de amostragem

O modo de alteração do tempo de amostragem encontra-se no fluxograma da figura 4.1-4. Esta função devolve o valor que alterou, porque para o estudo do sistema o tempo de amostra da velocidade é um dado importante, quer para a conversão do numero de impulsos lidos em rotações por minuto, quer na implementação de algoritmos de controlo.

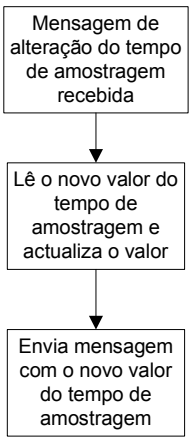


Figura 4.1-4 Fluxograma da alteração do tempo de amostragem

4.1.2.4 Activa ou desactiva o motor DC

O modo de alteração do estado do motor DC encontra-se na figura 4.1-5. Para que não se altere a velocidade de rotação do motor sem querer existe uma variável de activação ou desactivação do motor DC, se esta variável estiver desactiva estado lógico 0, o porto de saída correspondente à tensão do motor DC vai estar com o valor decimal 128 que corresponde ao nosso 0 volts do motor DC. Só depois de activar o motor DC é que é possível alterar o valor da tensão a aplicar ao motor.

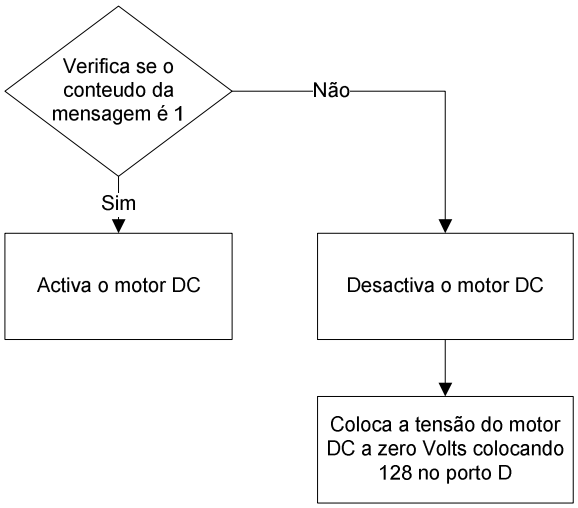


Figura 4.1-5 Fluxograma da alteração do estado do motor DC

4.1.2.5 Envia o valor da tensão do motor DC

O modo de alteração do valor de tensão a enviar ao motor encontra-se na figura 4.1-6, apenas é possível alterar a tensão do motor se este estiver activo.

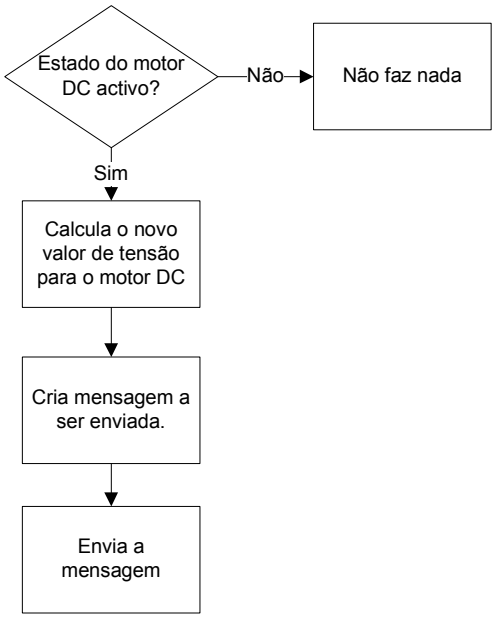


Figura 4.1-6 Fluxograma de alteração do valor da tensão

4.1.2.6 Envia o valor da velocidade do motor DC

A função que devolve o valor da velocidade do motor DC está na figura 4.1-7. O tempo de contagem de impulsos é feito pelo timer 2 e o tempo é dado pela variável tempo de amostragem, que pode ser alterado através de uma mensagem enviada pelo PC. A contagem de impulsos é efectuada pelo timer 1.

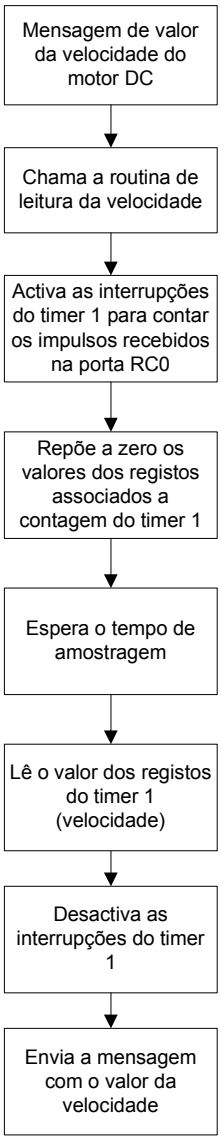


Figura 4.1-7 Fluxograma de medição de velocidade

O valor do registo do timer 1 T1CON é de 0x07 porque queremos activar o timer, ter o relógio externo que são os impulsos que aparecem no pino RC0 e não queremos sincronia com o relógio externo.

4.1.2.7 Envia o valor da corrente do motor DC

O comportamento da função que devolve o valor da corrente que é consumida pelo motor DC encontra-se na figura 4.1-8.

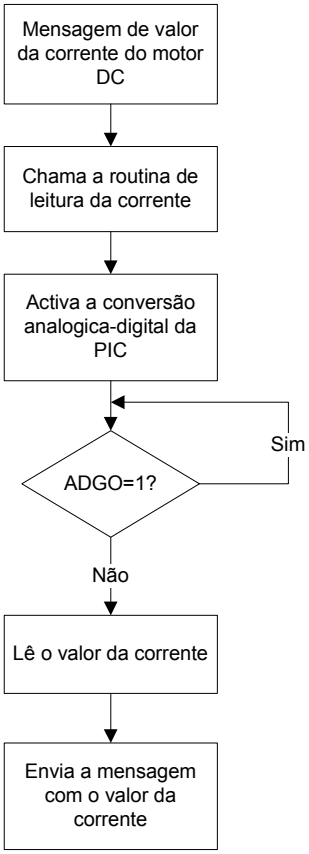


Figura 4.1-8 Fluxograma de medição de corrente no motor

4.1.2.8 Envia o valor da tensão do motor DC

O comportamento da função que devolve o valor da tensão aplicado ao motor DC encontra-se na figura 4.1-9.

Figura 4.1-9 Fluxograma de medição de tensão no motor

4.1.2.9 Altera o estado do motor passo-a-passo

O comportamento da função que altera o estado do motor passo-a-passo do estado activo para o estado desactivo, ou vice-versa, encontra-se na figura 4.1-10.

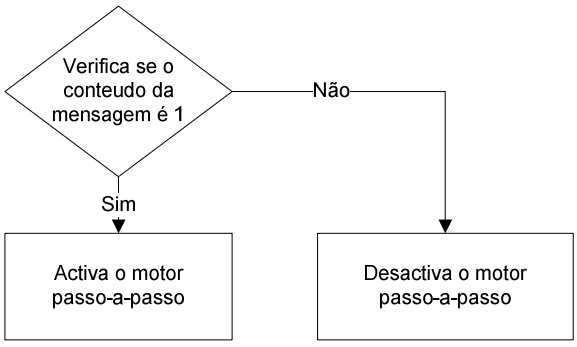


Figura 4.1-10 Fluxograma da alteração do estado do motor passo-a-passo

4.1.2.10 Altera o valor da velocidade do motor passo-a-passo

O comportamento da função que altera a velocidade do motor passo-a-passo encontra-se na figura 4.1-11. A velocidade do motor passo-a-passo é dada pela frequência do timer 0 que varia neste caso irá variar entre 76,29 Hz e 19,5 KHz. Para alterar a gama das frequências será necessário alterar o valor da divisão associado ao timer 0, no programa da PIC.

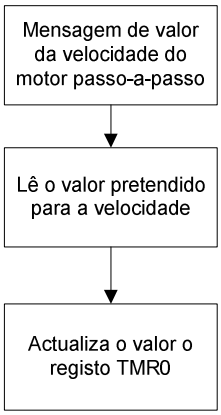


Figura 4.1-11 Fluxograma de alteração da velocidde no motor

4.1.2.11 Altera o sentido de rotação do motor passo-a-passo

O comportamento da função que altera o sentido de rotação do motor passo-a-passo encontra-se na figura 4.1-12.

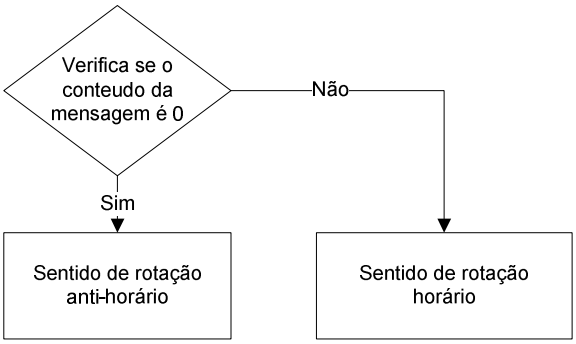


Figura 4.1-12 Fluxograma de alteração do sentido de rotação do motor passo-a-passo

4.1.2.12 Envia o valor da velocidade do motor passo-a-passo

O comportamento da função que envia a velocidade do motor passo-a-passo encontra-se na figura 4.1-13.

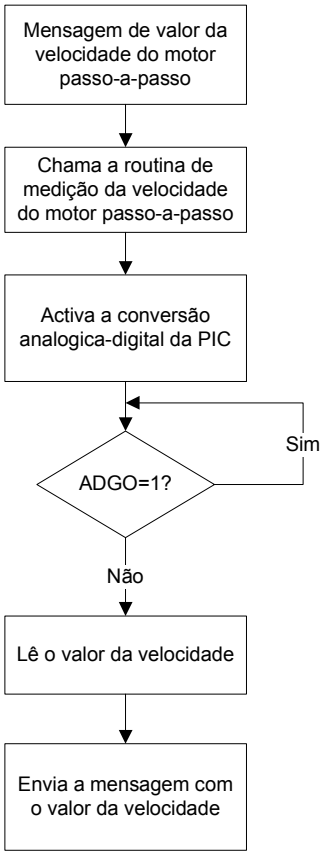


Figura 4.1-13 Fluxograma de medição de velocidade do motor passo-a-passo

4.1.2.13 Alterar o valor dos parâmetros do PID

Para controlar a velocidade de rotação do motor DC através do controlador PID implementado na PIC é necessário enviar os valores pretendidos para o Kp, Ki e Kd. O modo de alterar os valores de controlo é similar entre os 3, o que altera é a variável que vai ser alterada. O fluxograma de alteração do kp está na figura 4.1-14. Por defeito estes valores estão definidos a zero.

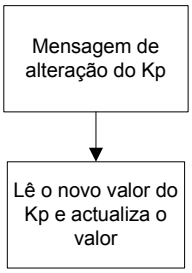


Figura 4.1-14 Alteração do Kp

4.1.2.14 Controlador PID

O fluxograma de controlo PID através da PIC encontra-se na figura 4.1-15.

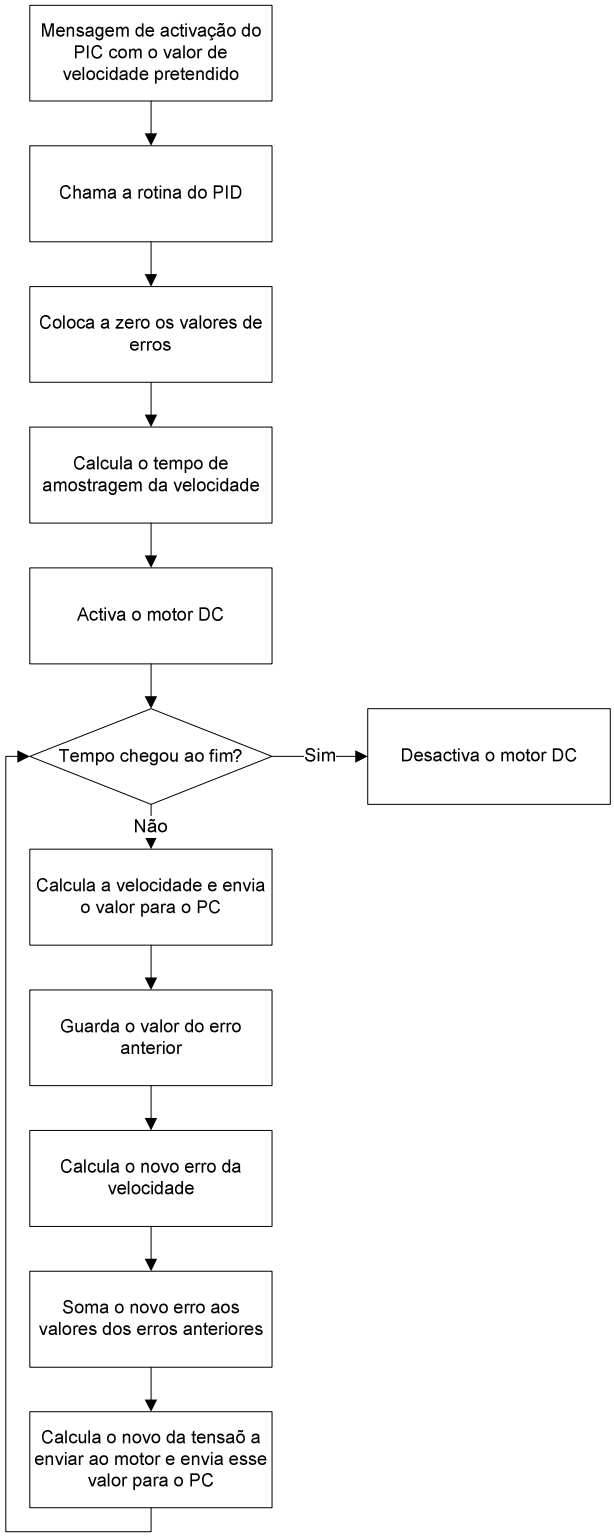


Figura 4.1-15 Controlo PID motor DC

4.2 Software em Matlab

Para facilitar a interacção entre a PIC e o utilizador, criou-se rotinas no Matlab que permitam fazer as operações mais elementares, tais como a leitura ou a escrita de mensagens. Depois foram criadas outras rotinas que permitem fazer a leitura da velocidade, corrente e a tensão, entre outras. As funções existentes e o seu modo de funcionamento encontram-se a seguir.

4.2.1 A função “Abrir_porta_serie”

A função “Abrir_porta_serie” do Matlab é uma função que dado o nome da porta série que pretendemos usar para a comunicação entre o PC e a PIC, vai em primeiro lugar testar se consegue abrir a comunicação, e se tiver tido sucesso em abrir a porta vai através da função “Test_Comu” verificar se consegue comunicar com a PIC através do envio de uma mensagem seguido da leitura da mensagem de resposta da PIC. Em caso de sucesso da abertura da porta de comunicação esta fica aberta e pronta para as comunicações que sejam necessárias para efectuar os testes que se pretendam realizar. Se não for possível abrir a porta não será possível comunicar com a PIC e comandar o sistema através do PC.

O fluxograma de funcionamento da função “Abrir_porta_serie” está representado na figura 4.2-1.

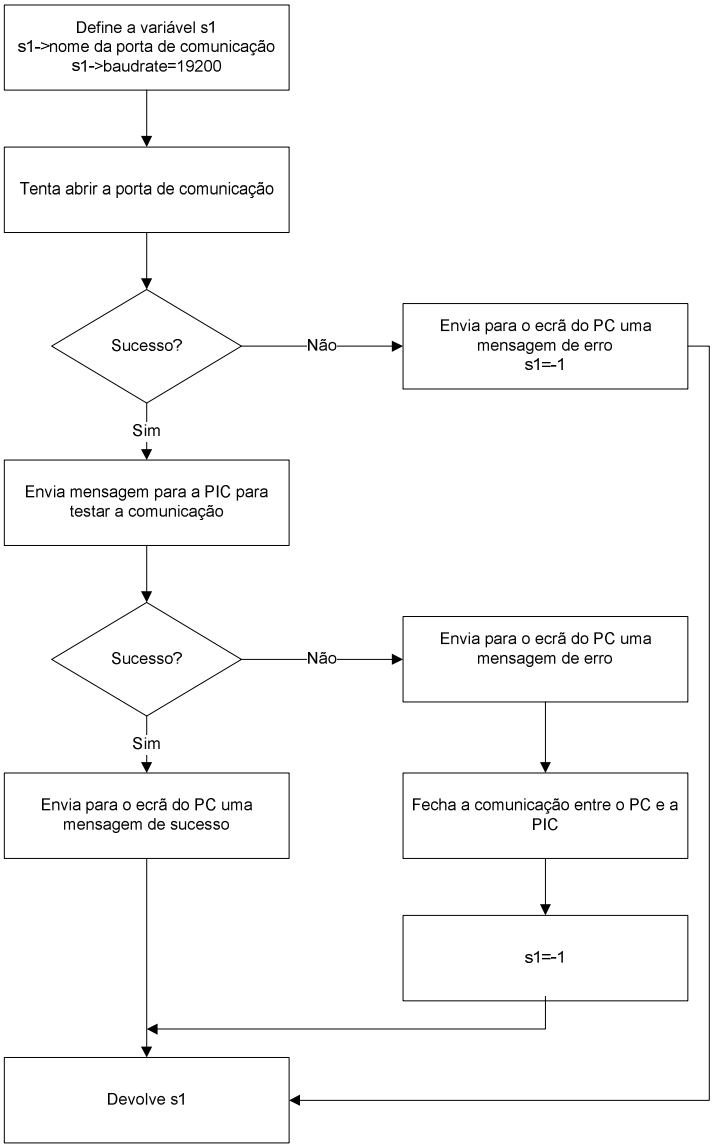


Figura 4.2-1 Função de abertura das comunicações

4.2.2 Função “Test_Comu”

O fluxograma de funcionamento da função “Test_Comu” está representado na figura 4.2-2.Esta função serve para testar as comunicações entre o PC e a PIC enviando uma mensagem de teste e verificando a mensagem de retorno.

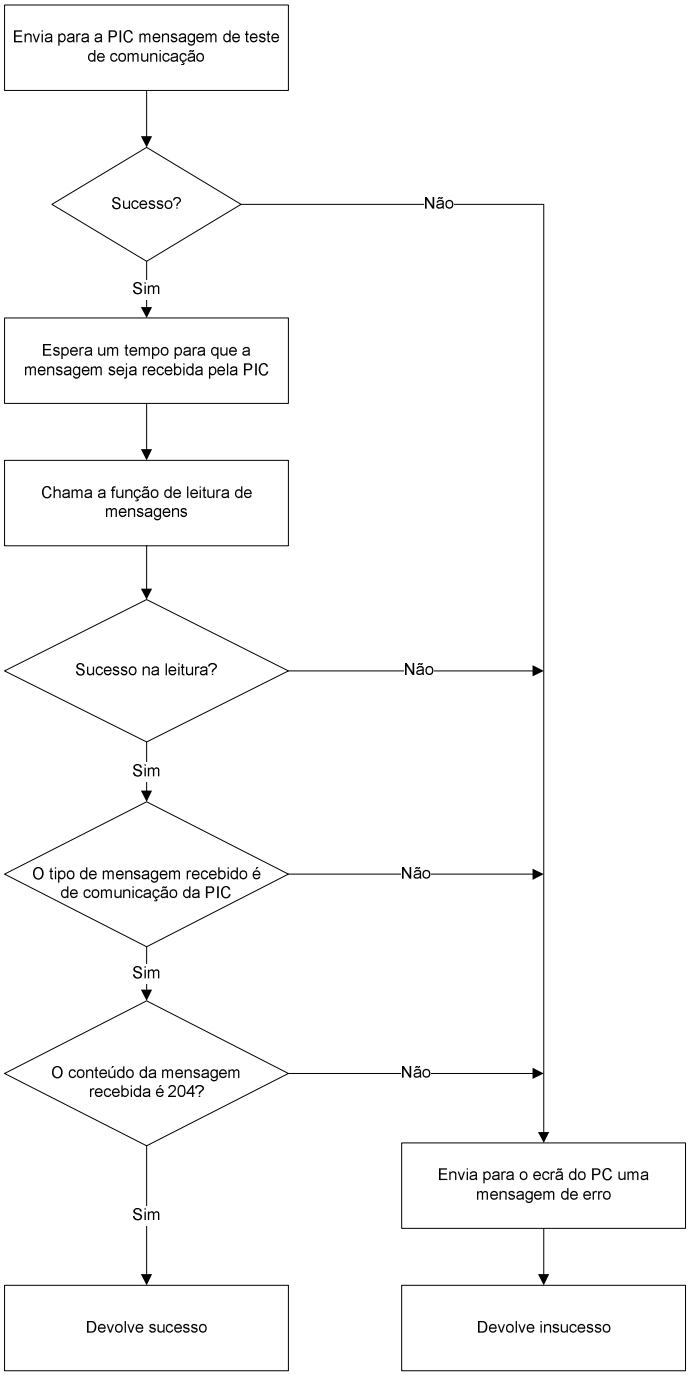


Figura 4.2-2 Função de teste das comunicações

4.2.3 A função “ENVIA_MSG”

A estrutura da função de envio de mensagem do PC, via Matlab, para a PIC encontra-se no fluxograma da figura 4.2-3. Se a mensagem a enviar tiver um valor inferior a 256, pode-se enviar o valor em apenas 8 bits, e optou-se por esta opção porque quanto menor for o tamanho da mensagem a enviar maior será a velocidade do envio da mensagem e mais rápido será o sistema global a responder. Tendo em conta que a rapidez de comunicação tem uma importância fundamental na implementação deste projecto, como se refere mais a frente no capitulo de teste da plataforma de controlo. Se o conteúdo da mensagem for maior que 256 torna-se necessário separar o valor a enviar por casas decimais, e verificar o tamanho do conteúdo a enviar para introduzir esse valor no corpo da mensagem a enviar.

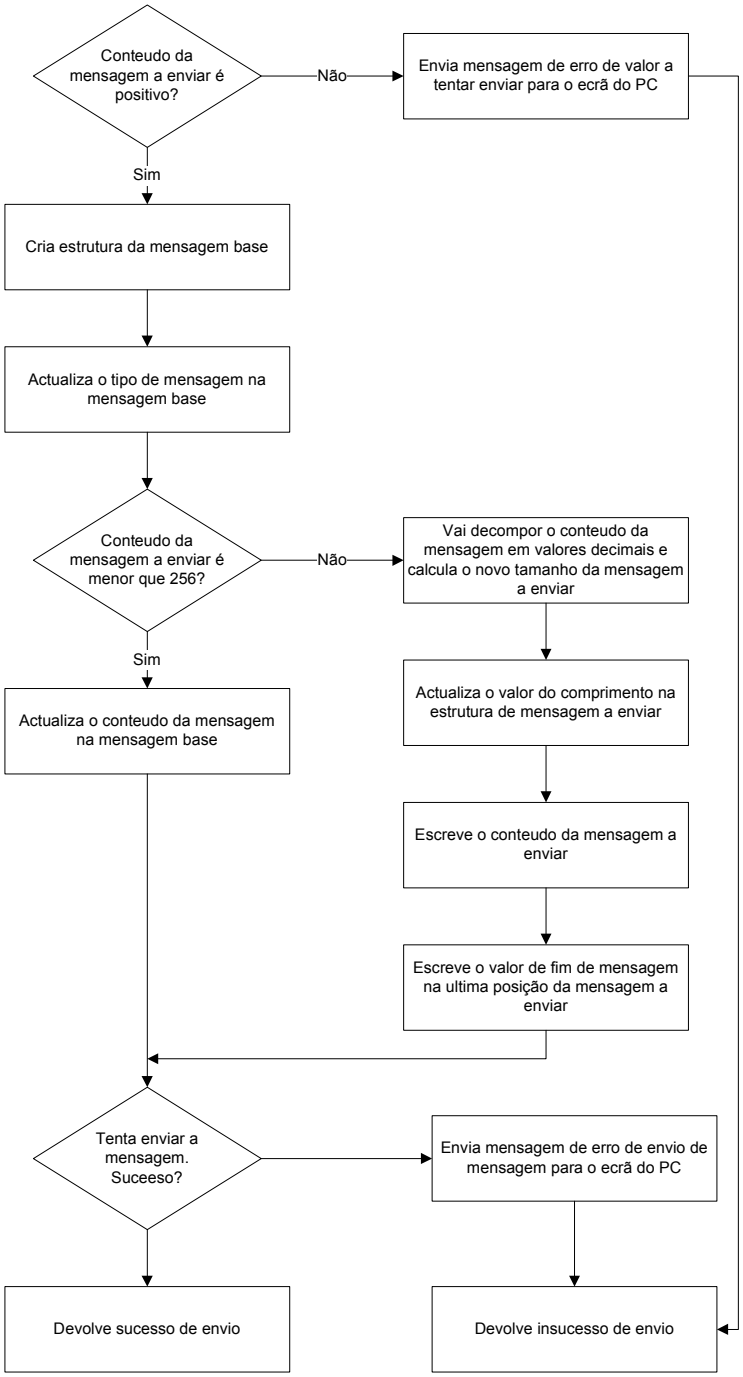


Figura 4.2-3 Função de envio de mensagens

4.2.4 A função “ LE_MSG”

A estrutura da função de recepção de mensagem do PC, via PIC, para o PC encontra-se no fluxograma da figura 4.2-4.

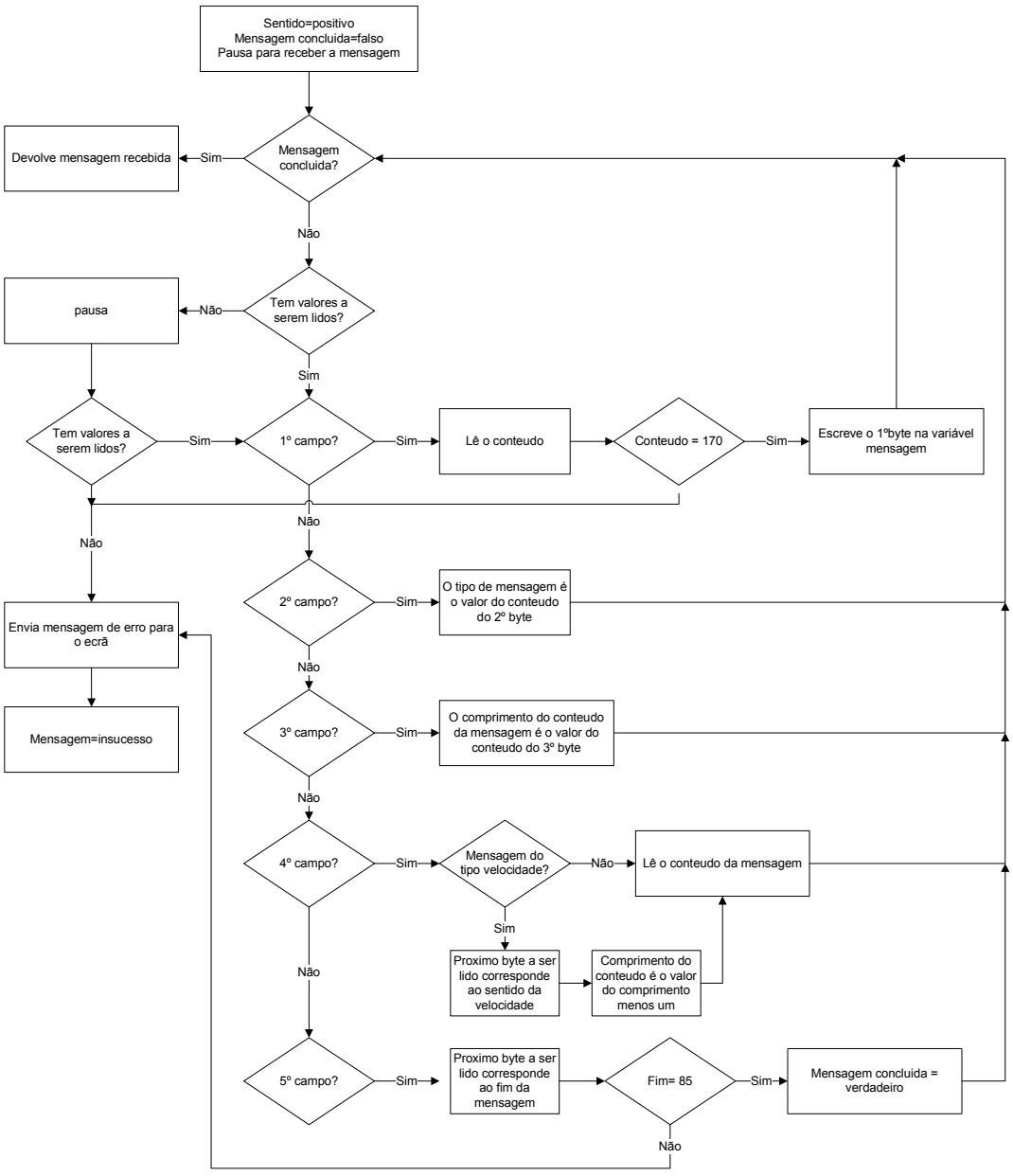


Figura 4.2-4 Função de leitura da mensagem recebida

4.2.5 A função “Tempo_Amost”

Para que se possa alterar o tempo em que a PIC está a ler os impulsos relativos à velocidade de rotação do motor DC, usa-se a função “Tempo_Amost” em que o valor do tempo de amostragem está ms. O fluxo de funcionamento desta função encontra-se na figura 4.2-5.

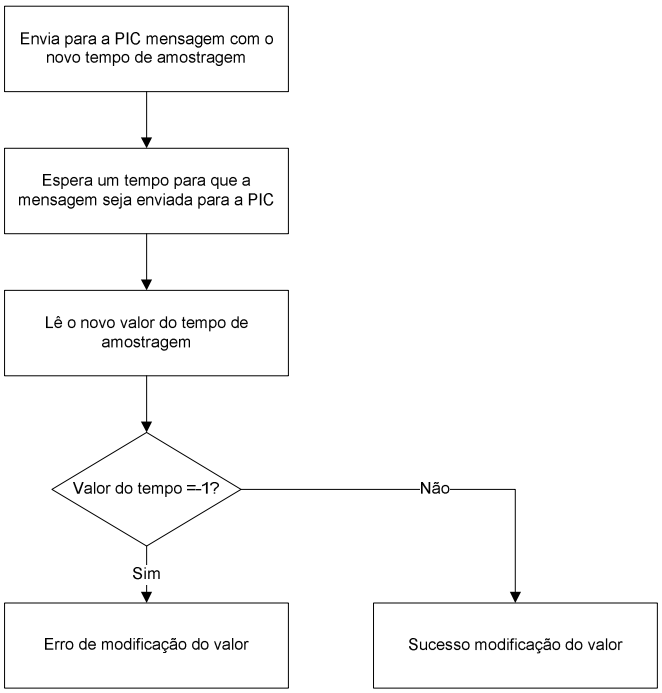


Figura 4.2-5 Função de alteração do tempo de amostragem

4.2.6 A função “Recebe_Corrente”

Para medir a corrente que o motor DC está a consumir usa-se a função “Recebe_Corrente” em que o valor da resolução tem de estar em mA. O valor enviado pela PIC é o valor digital da leitura, optou-se por fazer os cálculos da conversão digital para o valor analógico no Matlab para facilitar a comunicação, torna-se mais simples enviar valores inteiros do que valores fraccionais. E porque também é mais simples fazer contas com virgula flutuante no Matlab do que na PIC. Optou-se por ter uma resolução analógica para digital de 10 bits para poder ter uma maior precisão no valor medido. O fluxo de funcionamento desta função encontra-se na figura 4.2-6.

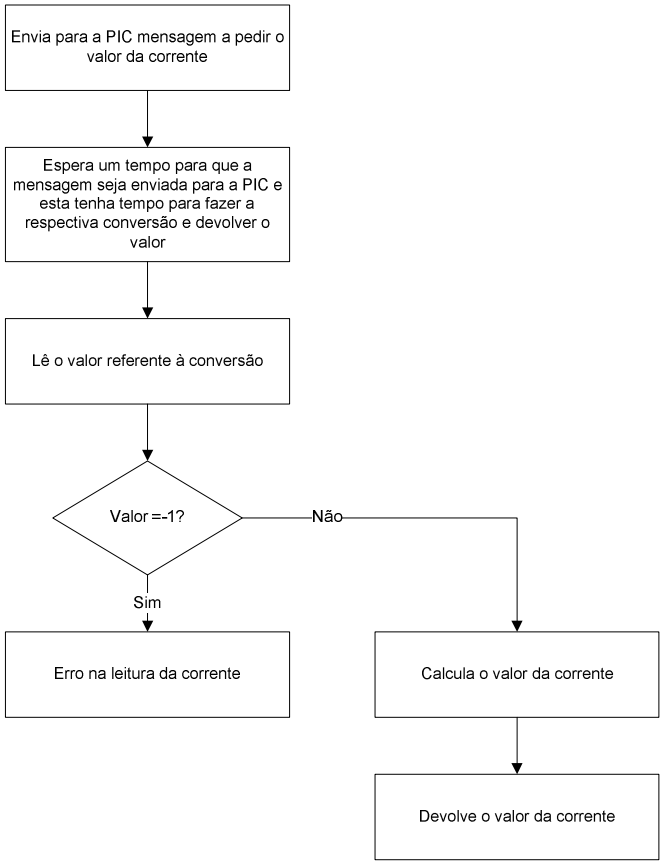


Figura 4.2-6 Função de leitura da corrente no motor DC

4.2.7 A função “Recebe_Tensao”

Para medir a tensão que está a alimentar o motor DC usa-se a função “Recebe_Tensao” em que o valor da resolução tem de estar em V. O valor enviado pela PIC é o valor digital da leitura, tal como na função de leitura de corrente. O fluxo de funcionamento desta função encontra-se na figura 4.2-7.

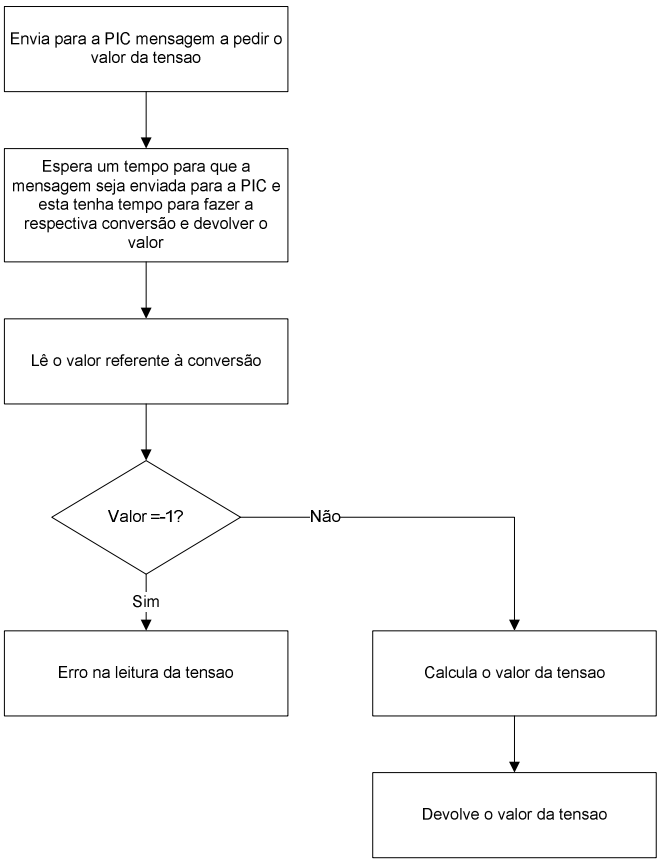


Figura 4.2-7 Função de leitura da tensão no motor DC

4.2.8 função “PID”

O modo de funcionamento da função do PID está descrito na figura 4.2-8.

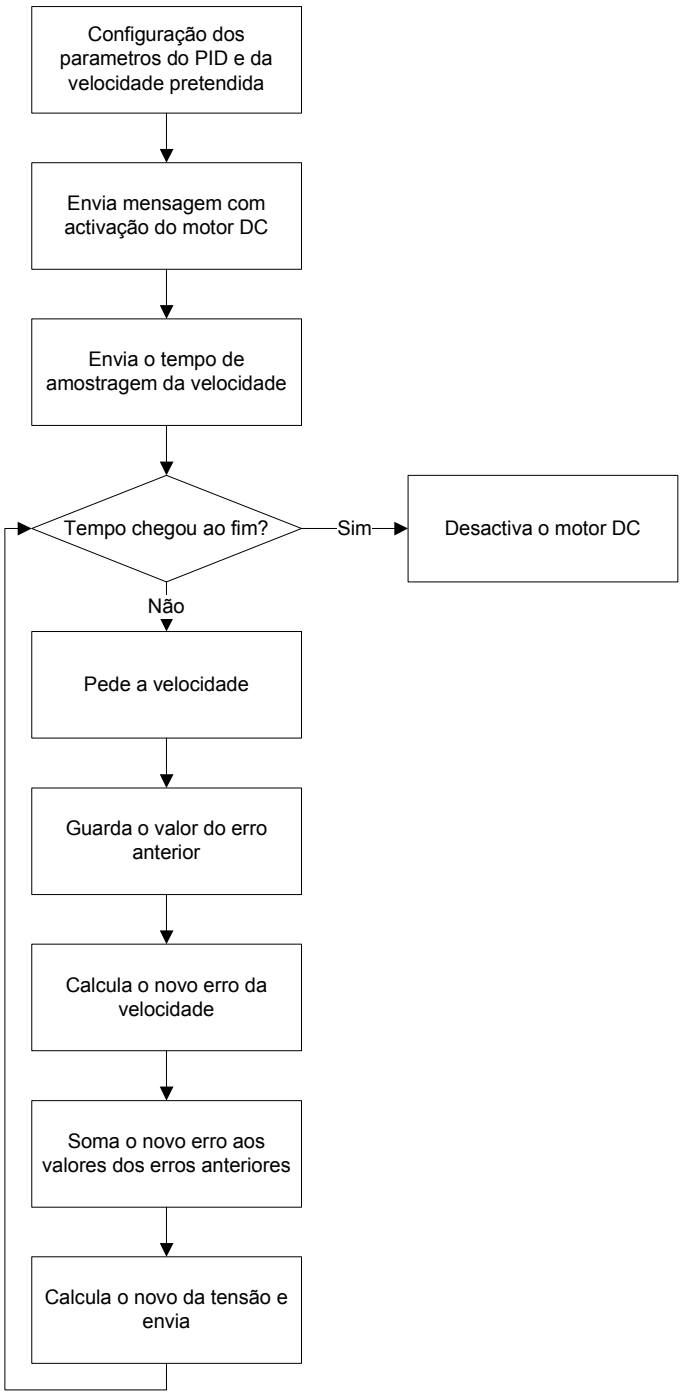


Figura 4.2-8 Controlo PID através do MatLab

5 Teste da plataforma de controlo de motores

Exemplos de gráficos obtidos com o MATLAB

5.1 Passo-a-passo como encoder

Na figura 5.1-1 está o gráfico resultante do estudo da velocidade do motor DC usando o motor passo-a-passo como *encoder*.

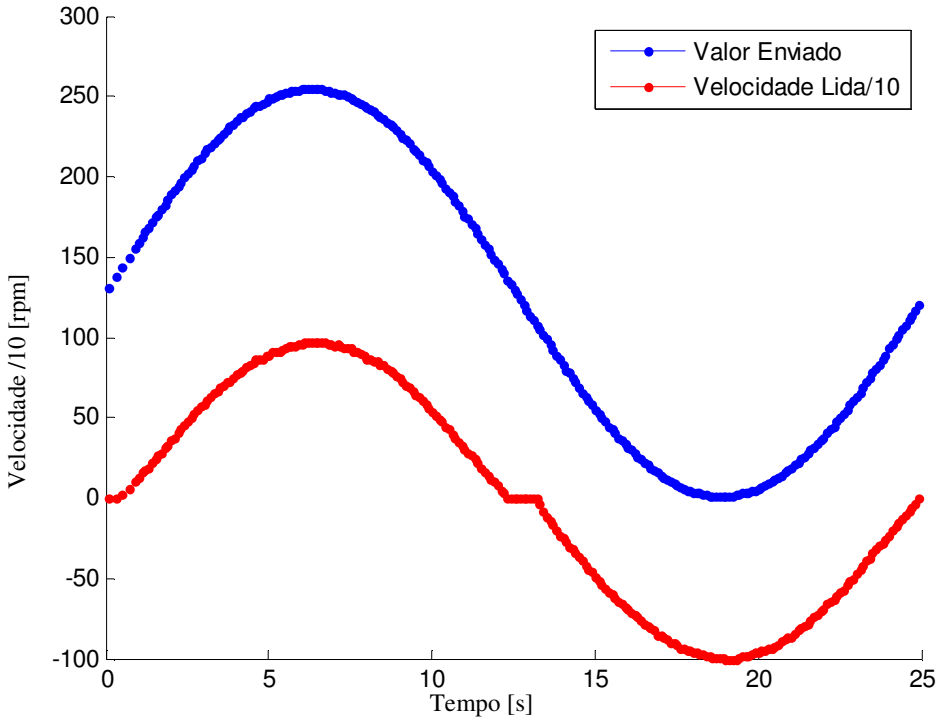


Figura 5.1-1 Velocidade motor

5.2 Estudo da velocidade do motor DC

Em todos os gráficos seguintes os pontos a vermelho correspondem a valor da velocidade em rpm, e os pontos azuis da "tensão" que estão a ser enviada ao motor. Este valor varia entre -128 e 128, este valor corresponde a um valor digital.

Nas figuras 5.2-1 e 5.2-2 temos aplicado um impulso com amplitude de 100 e 128, respectivamente. A amplitude é o valor digital que sai da PIC para a DAC.

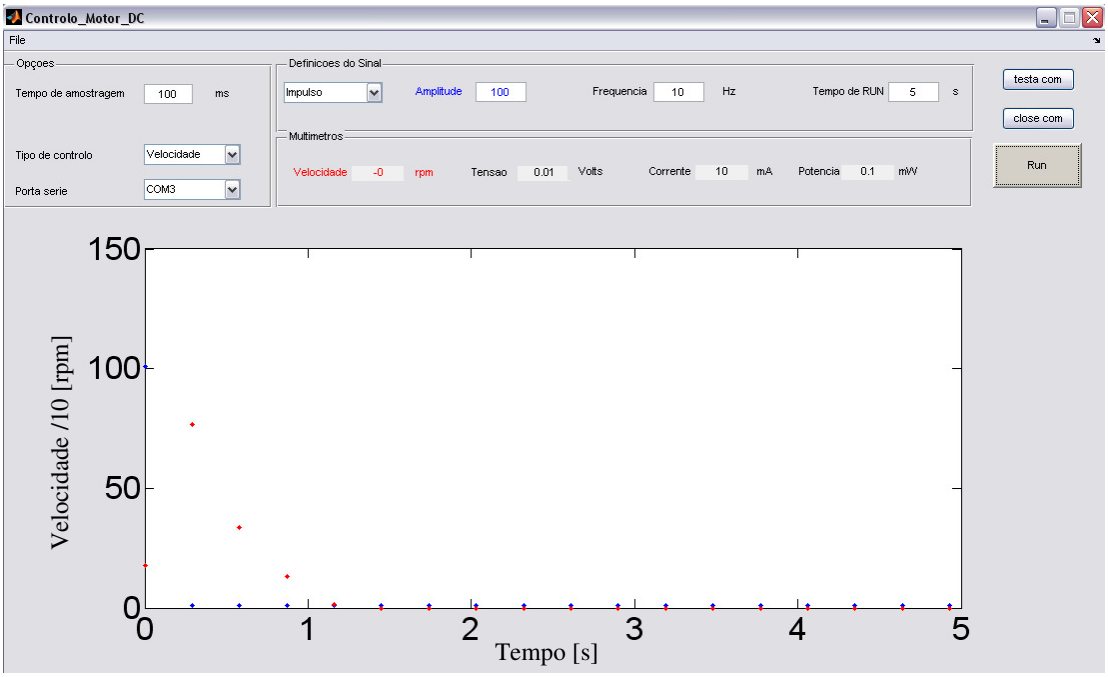


Figura 5.2-1 Exemplo da resposta a um impulso

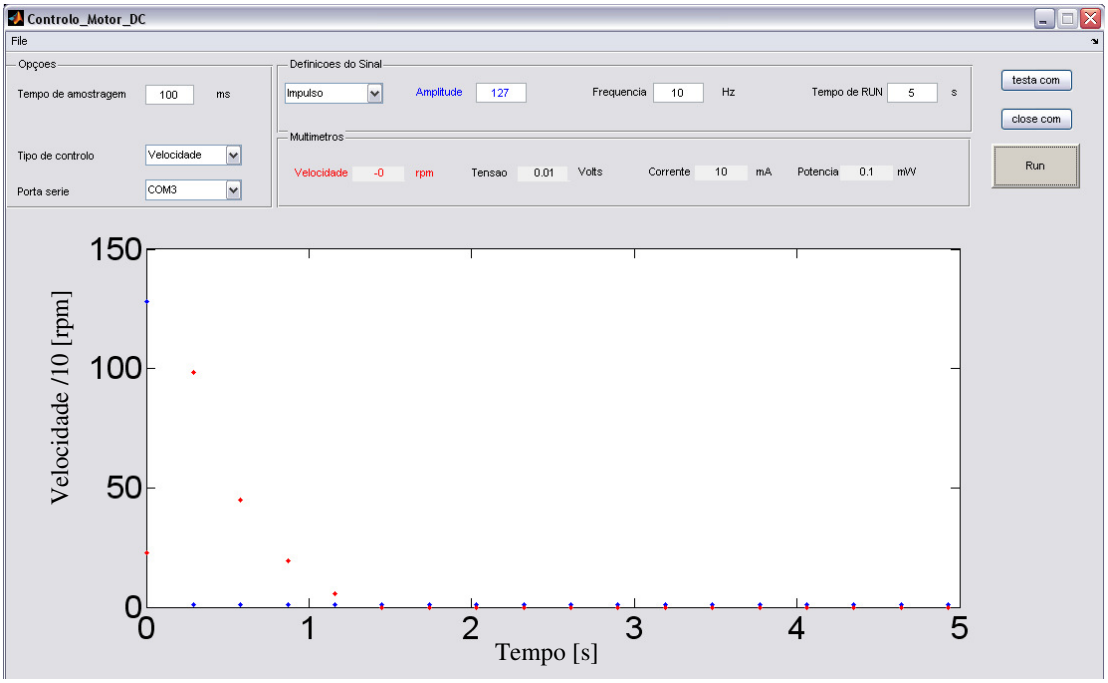


Figura 5.2-2 Resposta a um impulso

Nas figuras em baixo estão representados vários sinais, em vários instantes. O tempo de amostragem, o tipo de sinal, o valor máximo da amplitude, o tempo de execução e os valores instantâneos medidos, podem ser vistos nas figuras.

Na figura 5.2-3 está a resposta a um seno com uma frequência de 0.1 Hz e uma amostragem de velocidade a cada 100 ms. Na figura 5.2-4 está a resposta a um seno também com uma frequência de 0.1 Hz e uma amostragem de velocidade a cada 10 ms. E na figura 5.2-5 o mesmo seno para um tempo de amostragem de 1ms.

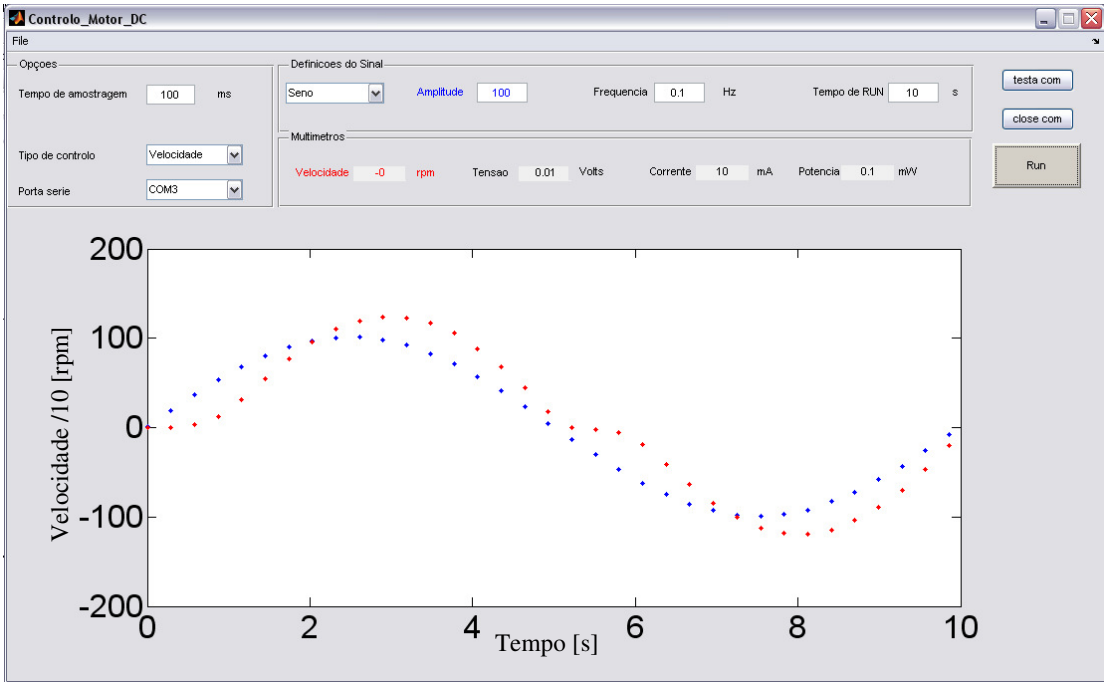


Figura 5.2-3 Resposta a um seno

Título da Dissertação:
Sistema servo-mecanismo didáctico com motores DC e passo-a-passo.

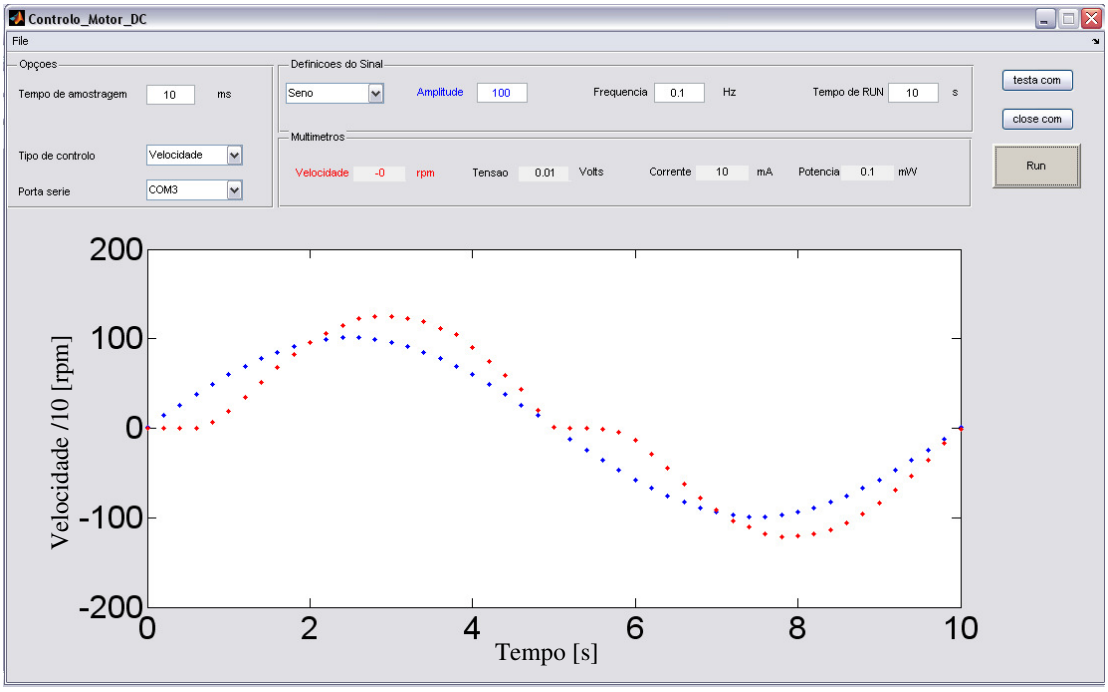


Figura 5.2-4 Resposta a um seno

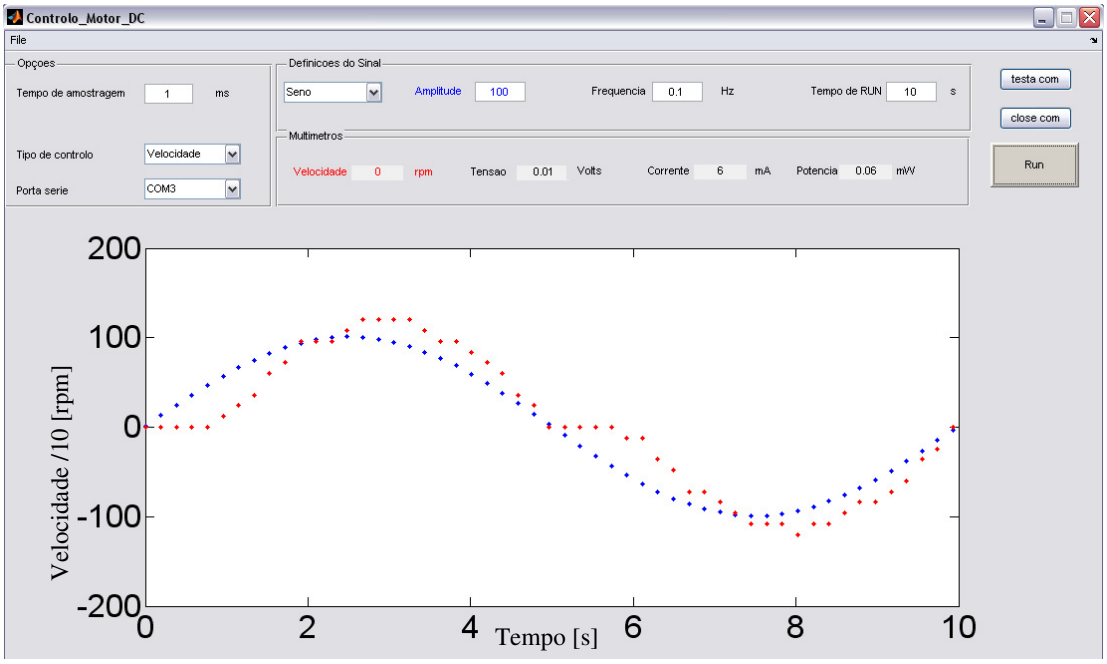


Figura 5.2-5 Resposta a um seno

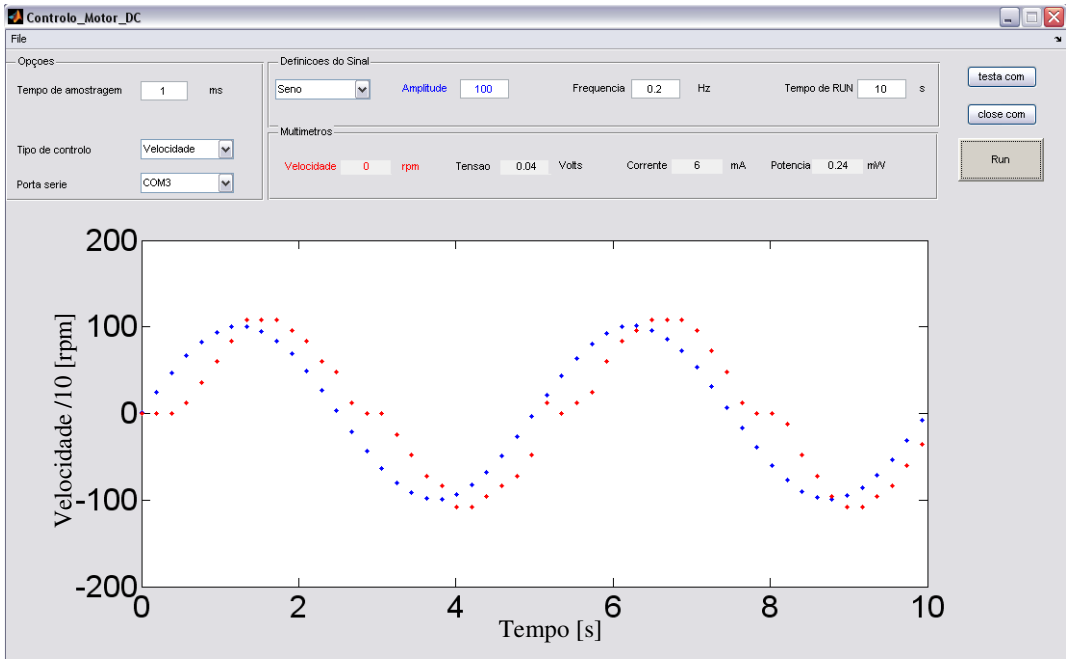


Figura 5.2-6 Resposta a um seno

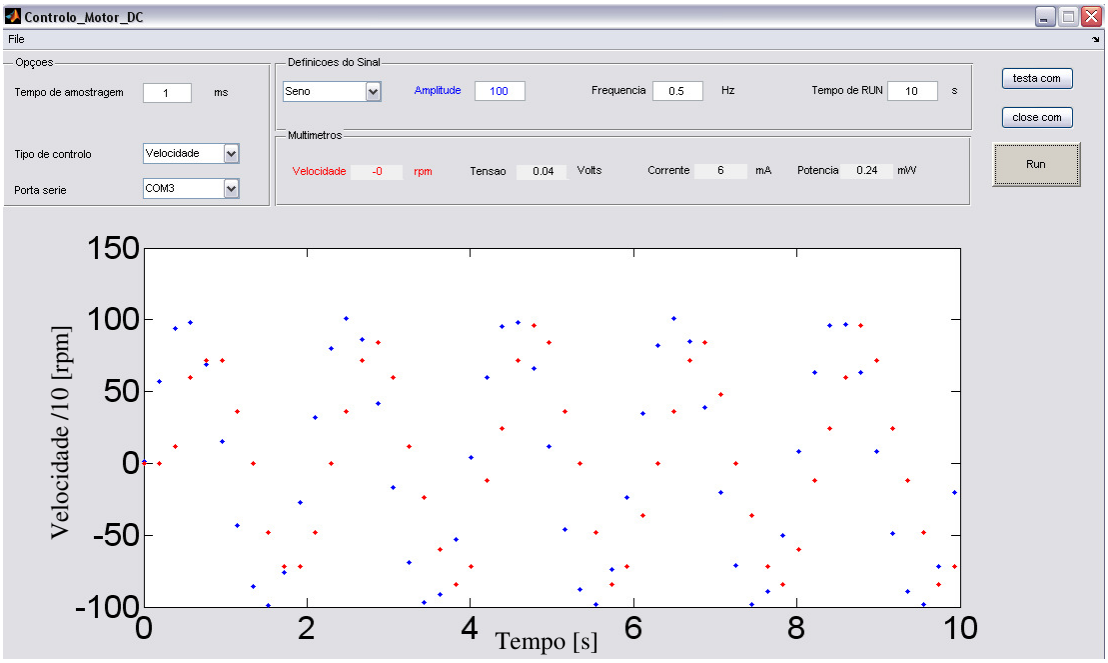


Figura 5.2-7 Resposta a um seno

Título da Dissertação:
Sistema servo-mecanismo didáctico com motores DC e passo-a-passo.

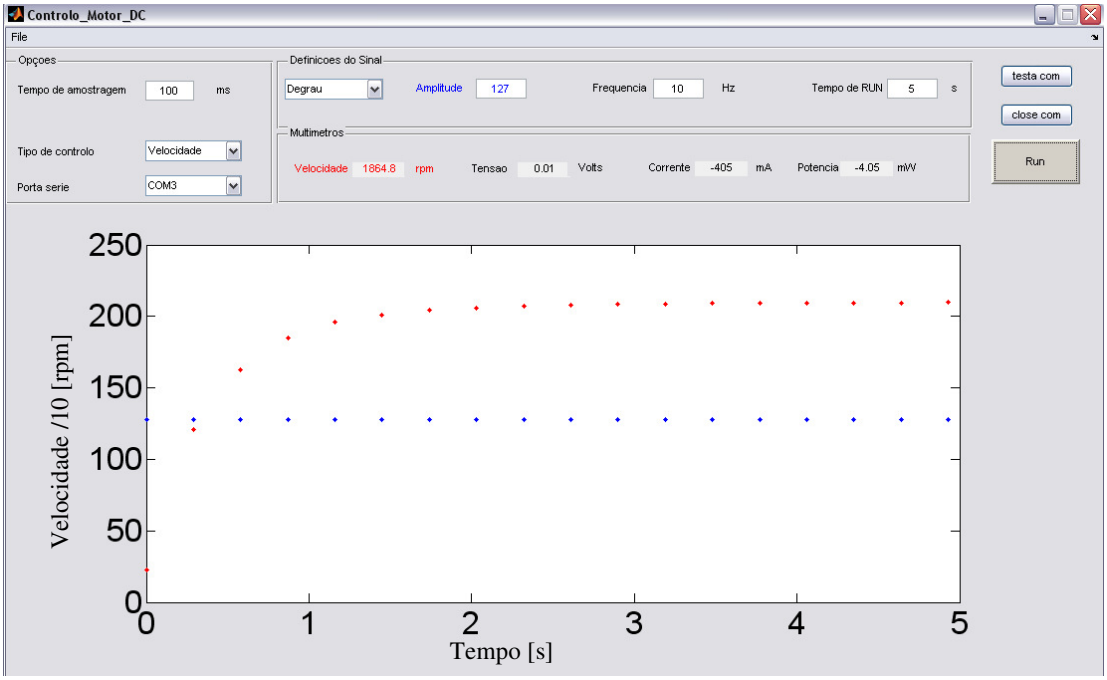


Figura 5.2-8 Resposta do sistema a um degrau com amplitude 127

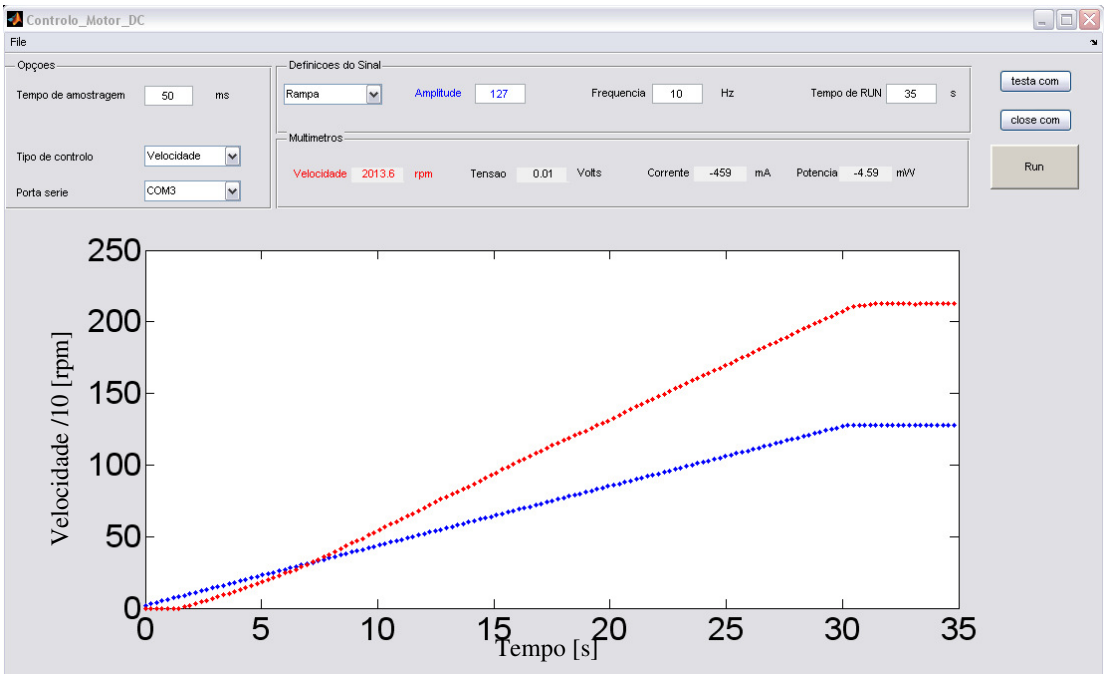


Figura 5.2-9 Resposta do sistema a uma rampa

5.3 Controlador PID implementado no MATLAB

Foram testados alguns valores de Kp, Ki e Kd, as suas respostas encontram-se nas figuras seguintes.

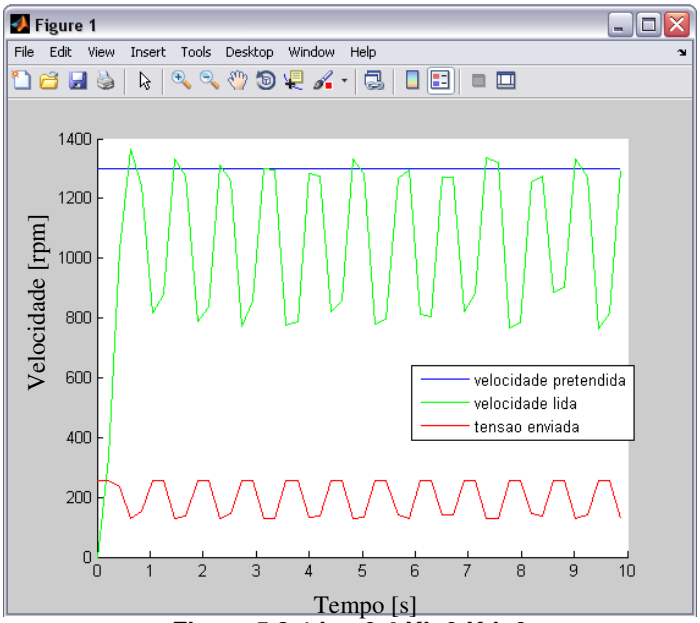


Figura 5.3-1 kp=0.4 Ki=0 Kd=0

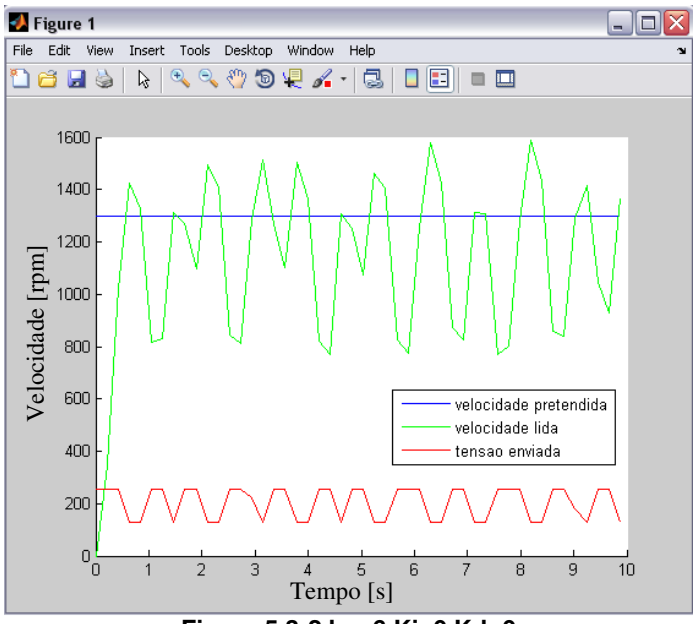


Figura 5.3-2 kp=6 Ki=0 Kd=0

Como se pode ver através da análise das figuras 5.3-1 e 5.3-2 quanto maior for o valor do Kp mais o sinal se aproxima do valor desejado, embora apenas com este controlador o sistema tem muitas oscilações. Também se pode verificar que o controlo através do MatLab tem muitas falhas porque está dependente dos tempos de comunicação e quando o valor é enviado o sistema

este já não tem necessariamente a mesma velocidade que fora enviada num instante anterior ao PC, o que faz com que as correcções que tem que ser efectuadas sejam mais bruscas e o resultado é o que se pode observar nas figuras.

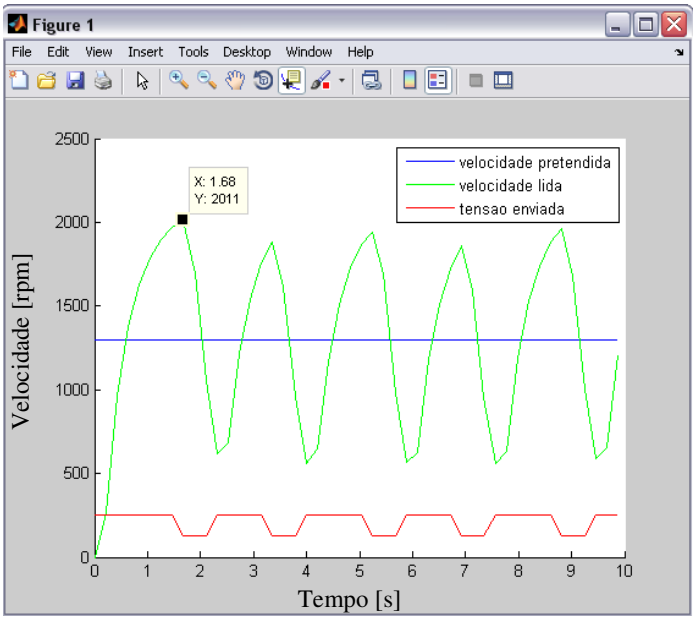


Figura 5.3-3 kp=20 Ki=5 Kd=10

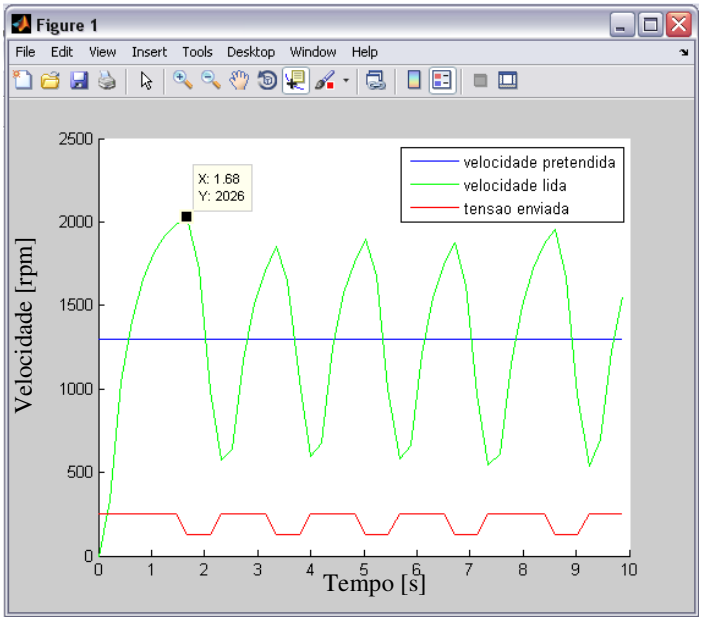


Figura 5.3-4 kp=20 Ki=5 Kd=20

Pela análise das figuras 5.3-3 e 5.3-4 verifica-se que o sistema controlado pelo MatLab é muito instável e que não dá para retirar conclusões.

5.4 Controlador PID implementado na PIC

Gráficos obtidos com o controlador da PIC para diferentes valores de Kp, Ki e Kd estão demonstrados em baixo, a velocidade pretendida era de 1000 rpm.

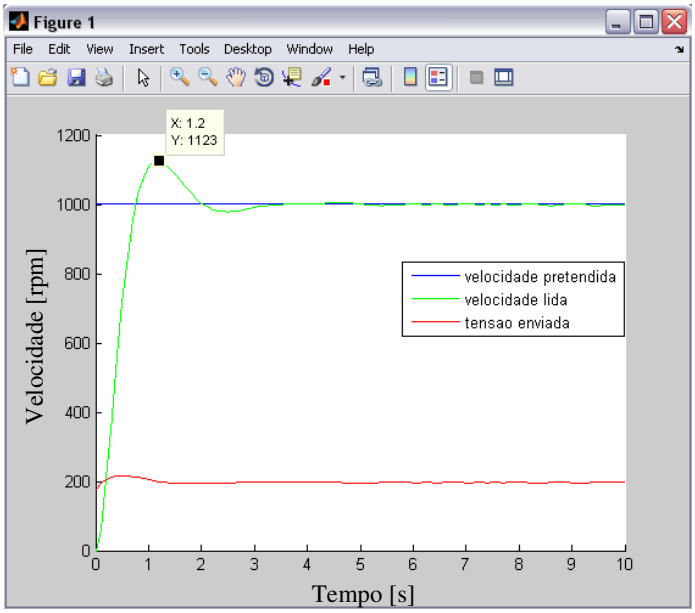


Figura 5.4-1 kp=0.03 Ki=0.0002 Kd=0.0005

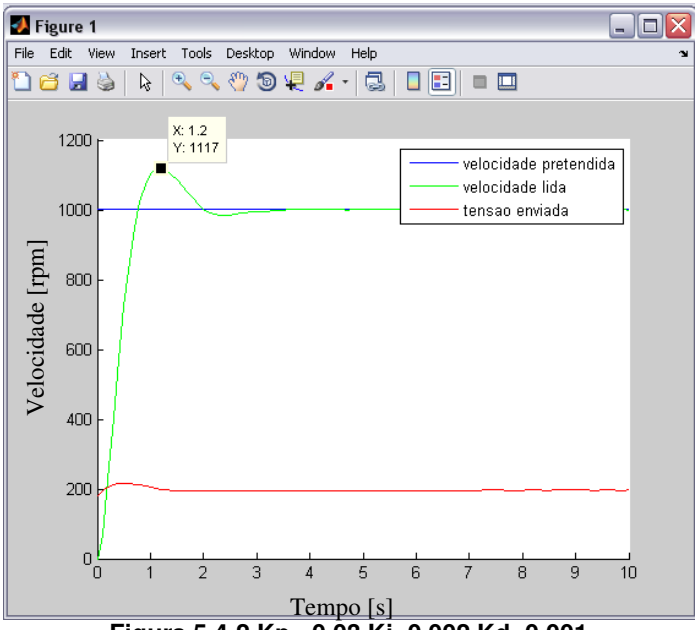


Figura 5.4-2 Kp= 0.03 Ki=0.002 Kd=0.001

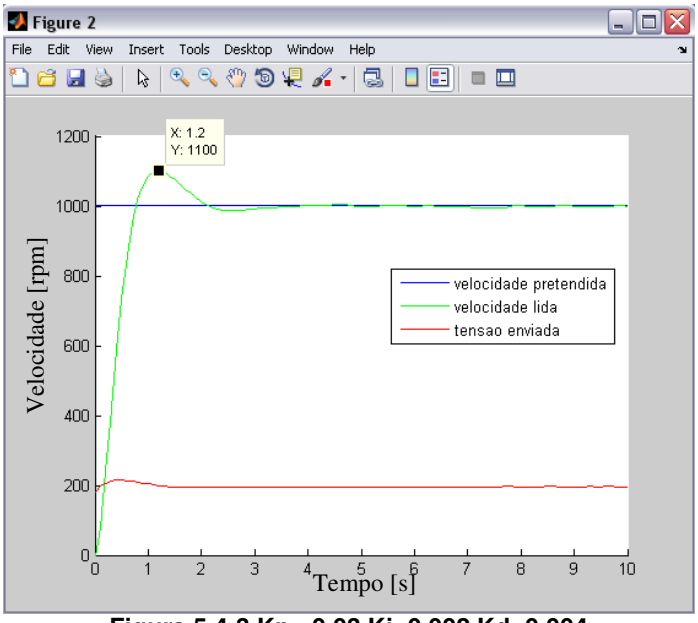


Figura 5.4-3 Kp= 0.03 Ki=0.002 Kd=0.004

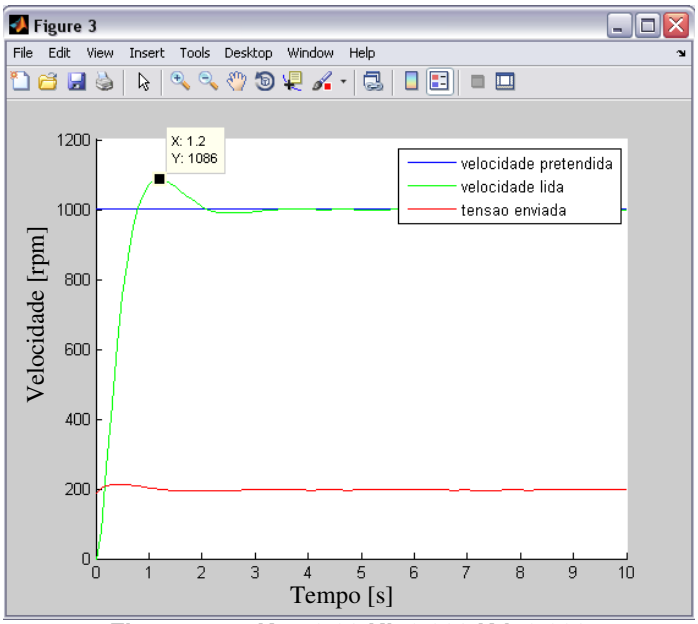


Figura 5.4-4 Kp= 0.03 Ki=0.002 Kd=0.008

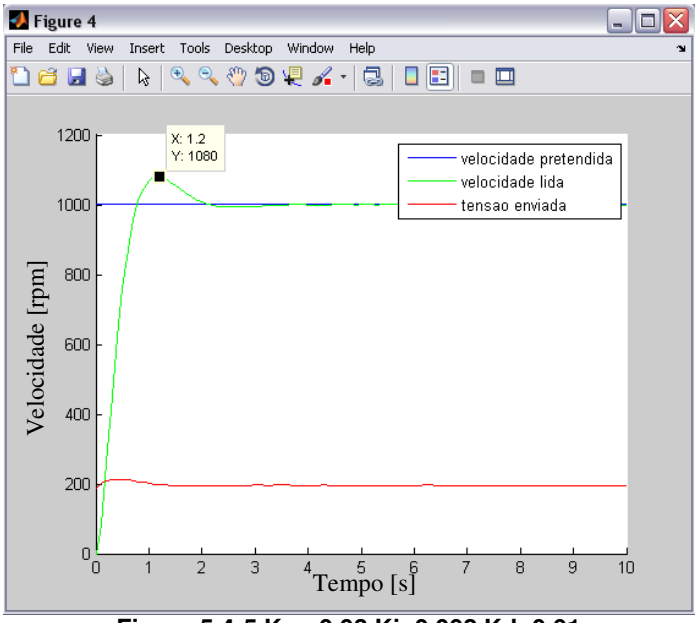


Figura 5.4-5 Kp= 0.03 Ki=0.002 Kd=0.01

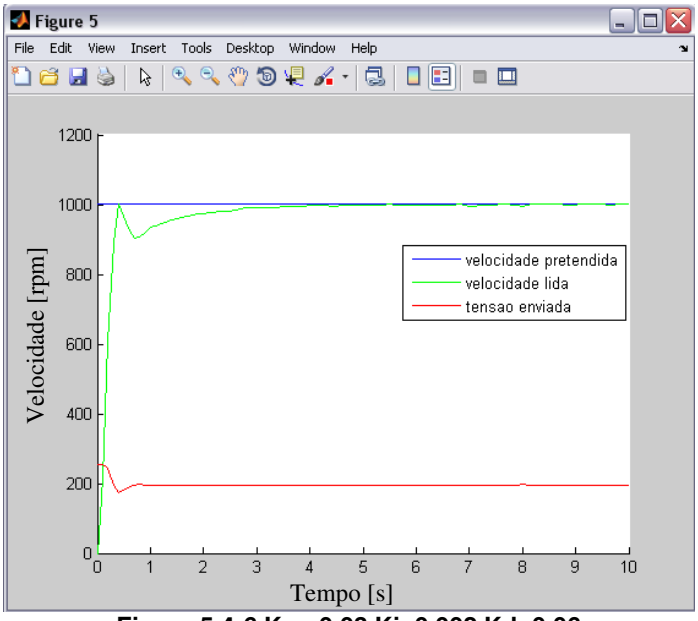


Figura 5.4-6 Kp= 0.03 Ki=0.002 Kd=0.06

Como se pode ver através das figuras 5.4-1 a 5.4-6 quanto maior for o valor de Kd menor é a sobre elevação, mas a partir de um determinado valor o gráfico obtido começa a degenerar-se.

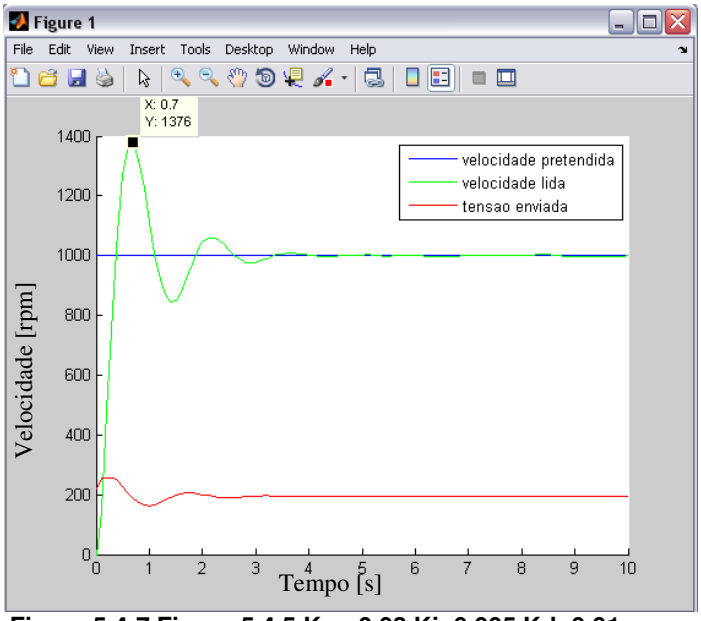


Figura 5.4-7 Figura 5.4 5 Kp= 0.03 Ki=0.005 Kd=0.01

Como se pode ver através das figuras 5.4-5 e 5.4-7 ao aumentar o valor do Ki o tempo de subida diminui, mas a sobre elevação aumenta bastante, como seria de esperar.

Como se pode comparar entre este estudo e o do anterior, implementação do PID no MatLab este tem uma melhor resposta, sendo muito mais rápido que o anterior.

6 Conclusões e trabalho futuro

Verificou-se que o tempo de resposta do sistema é muito lento quando em uso com o MatLab. Seria de esperar para o motor DC uma função de transferência de 2ª ordem e como tal a resposta ao degrau deveria ser uma curva com uma forma de "S" o que não foi possível observar tendo em conta que o tempo mínimo de intervalo de amostras é de cerca de 100 ms. Em que este intervalo conta com os tempos de envio de mensagens, como o envio da tensão a aplicar e da mensagem a pedir a medição da velocidade de rotação, o tempo que demora a fazer a contagem de impulsos e a enviar a mensagem como o valor dos impulsos. A soma de todos os tempos que o sistema leva a processar as mensagens a executar os códigos necessários faz com que a escolha feita para a taxa de transmissão se tenha vindo a verificar insuficiente, para futuro trabalho é necessário aumentar a taxa de transmissão e verificar se é possível medir a resposta ao degrau com amostragens suficientes para se obter a função de transferência em segundo grau do sistema.

Como não foi possível saber as características mecânicas dos motores por estes não terem o modelo ou qualquer referência que pudesse levar as características de construção dos mesmos, será necessário fazer um estudo do sistema por forma a medir as características físicas dos mesmos tais como a resistência do motor a impedância e os respectivo binário motor.

Para o motor passo-a-passo será necessário fazer todo o estudo dinâmico e de controlo. O controlo de velocidade do motor passo-a-passo precisa de ser melhorado a nível de software.

7 Bibliografia

[1] http://www.electronica-pt.com/index.php/content/view/202/37/

[2] http://electronics.howstuffworks.com/motor.htm

[3] http://www.siemens.com.br/medias/FILES/2910_20060505141908.pdf

[4] http://homepages.which.net/~paul.hills/SpeedControl/SpeedControllersBody.html

[5] Microelectronic Circuits - Sedra Smith - 5th edition capitulo 14 secção 14.3.5

[6] “Andares de saída e amplificadores de potência” da cadeira de electrónica 2

[7] http://www.mspc.eng.br/eletrn/ampcl110.shtml#clas_a

[8] http://www.pc-control.co.uk/incremental_encoders.htm

[9] datasheet MCT7800

[10] MELO A. P.: Apontamentos teóricos de Sistemas de Controlo (2007), Cap.8.

8 Anexos

8.1 Listagem do código da PIC

```
//////////////////////////////////////////////////////
//                                                  //
//        Controlador do motor                      //
//                                                  //
//////////////////////////////////////////////////////

#include <htc.h>
#include "delay.h"


void interrupt RSI(void);
void recebe_mensagem(void);
void get_valor (void);

void comunicacao(void);
void velocidade (void);
void corrente (void);
void tensao (void);
void PID (int  vel_pret);
void velocidade_PP (void);

void cria_mensagem(unsigned int val,unsigned char type);
void envia_mensagem(unsigned char i, unsigned int *);
unsigned char bin2asc(unsigned char num);

//Variaveis globais

#define MOTOR_PP                RC3
#define SEN_MOTOR_PP            RC4
#define MOTOR_PP_POWER          RC2

unsigned int Count=0;
unsigned int freq;
unsigned int tempo, tempo_cont;         // TEMPO DE AMOSTRAGEM DE IMPULSOS
unsigned int Flag_R, rec, sentido;
unsigned int str_R[10];
unsigned int Power_PP=1;
unsigned char ok;
unsigned char MOTOR_DC;
float kp, ki, kd;
unsigned long Tempo_PID=10000;      // tempo de actuação do motor DC ms


//*****************************************************************************
// configuracao da Usart e portos
//
void CONFIG (void)
{
        // programacao dos portos

/* configurar ADC */
        ADCON1=0b10000100;                  /* right justified, 1 analog inputs pag 113/114*/
```

```
        ADCON0=0b10000001;              /* fosc/32, channel 0, ADON */
        ADIF=0;
        ADIE=0;                     /* A/D converter interrupt enable */

        TRISA=0b00001110;                       // input:RA1 RA2
        TRISB=0b00000010;                       // Input: RB1
        TRISC=0b10000001;               // ENTRADAS RC7->RX RC0->T1CK1 SAIDAS RC6->TX
                                        // RC2 clock   RC3 enable  RC4 sentido
        TRISD=0b00000000;               // Outport

        TMR0IF = 0;
        TMR0IE = 0;

        Power_PP=1;
        OPTION = 0b11010110;
        TMR0 = MOTOR_PP;

        RC2=0;
        // Programacao da Usart
        SPBRG = 64;                     //  BR=19200
        BRGH = 1;                       //  BR high speed
        SPEN = 1;                       //  Serial Port Enable
        TXEN = 1;                       //  Transmit Enable
        CREN = 1;                       //  Continuous Reception Enable
        RCIE=1;                         //      Receive Interrupt Enable bit

        PEIE=1;                         //      Peripheral interrupt enable
        GIE=1;                          //      Global interrupt enable

        freq=0;                         //  Impulsos medidos referentes a velocidade
        tempo=20;  // TEMPO DE AMOSTRAGEM DE VELOCIDADE EM tempo/0.05mS
        Flag_R=0;                       //      FLAG DE MENSAGEM RECIDA
        rec=0;                          //      BYTES RECEBIDOS
        ok=0;
        PORTD=128;                      //  Velocidade zero do motor
        MOTOR_DC=0;                     //      Motor DC off
        MOTOR_PP=0;                     //      Motor passo-a-passo off
        SEN_MOTOR_PP=1;             //  Sentido positivo
float kp=0.00;
float ki=0.00;
float kd=0.00;
}


void main (void)
{
        CONFIG();// Configuracoes das portas usart e timer 1 e variaveis de condições iniciais

        while(1){
                if(Flag_R==1)
//SE EXISTE UMA MENSAGEM COMPLETA RECEBIDA PROCESSA-A
                {
                        get_valor();
                        Flag_R=0;
                }
        }
}
```


Título da Dissertação:

Sistema servo-mecanismo didáctico com motores DC e passo-a-passo.

```
//*****************************************************************************
//
//                  FUNCAO DE ATENDIMENTO A INTERRUPCAO DA USART
//
//*****************************************************************************

void interrupt    RSI(void)
{
        unsigned char valor;

        GIE=0;
        PEIE=0;                                         //global interrupt disable
        if(RCIF != 0)
        {
                RCIE=0;                                         //disable interrupt
                recebe_mensagem();
                RCIE=1;                                         //enable interrupt
        }
                                                                // global interrupt enable

        if (TMR0IF==1 )
        {
           TMR0IF = 0;                          // Clear flag
           TMR0 = Power_PP;             // Reload TMR0 reg
                if(RC2==0)
                        RC2=1;
                else
                        RC2=0;
        }

        if (TMR2IF==1 )
        {
                TMR2IF=0;
           T2CON=0b01111100;
                PR2=0x0D;
                TMR2=0;
                Count++;
                if (Count==(1*tempo))
                {
                        Count=0;
                        ok=1;
                }
        }

        GIE=1;                                          // global interrupt enable
        PEIE=1;
}
//*****************************************************************************
//
//                  FUNCAO DE PROCESSAMENTO DA MENSAGEM RECEBIDA
//
//*****************************************************************************
void recebe_mensagem(void)
{
        unsigned int comp,i;

        str_R[rec++]=RCREG;
```

```
        if (rec==3)
                comp=str_R[2];

        if(rec==(comp+4))
        {
                Flag_R=1;
                rec=0;
                comp=0;
        }
}

//*****************************************************************************
//
//              FUNCAO QUE DESCODIFICA MENSAGENS
//
//*****************************************************************************
void get_valor (void)
{
        unsigned int final_val, comp, i, tipo;
        unsigned char valor;

        tipo=str_R[1];
        i=0;

        switch(tipo)
        {
                //Verifica comunicacao
                case 0X10:
                        comunicacao();
                        break;

                //Altera o tempo de leitura
                case 0X11:
                        comp=str_R[2];
                        final_val=0;
                        while (comp!=i)
                        {
                                valor=str_R[i+3];
                                final_val=10*final_val+valor;
                                i++;
                        }
                        tempo=final_val/0.05;
                        cria_mensagem(final_val,0x01);
                        break;

                //Altera o estado do motor
                case 0x12:
                        if (str_R[3]==1)
                                MOTOR_DC=1;
                        else
                        {
                                MOTOR_DC=0;
                                PORTD=128;
                        }
                        break;

                //Altera a velocidade
                case 0x13:
```

```
            if (MOTOR_DC==1)
            {
                        comp=str_R[2];
                        final_val=0;
                        while (comp!=i)
                        {
                                    valor=str_R[i+3];
                                    final_val=10*final_val+valor;
                                    i++;
                        }
                        PORTD=final_val;
            }
            break;

//Pede a velocidade
case 0X14:
            velocidade();
            cria_mensagem(freq,0x04);
            break;

//Pede a corrente
case 0X15:
            corrente();
            break;

//Pede a tensao
case 0X16:
            tensao();
            break;

//PID
case 0X17:
            comp=str_R[2];
                        final_val=0;
                        while (comp!=i)
                        {
                                    valor=str_R[i+3];
                                    final_val=10*final_val+valor;
                                    i++;
                        }
            PID(final_val);
            break;

//Enable/disable do motor passo-a-passo
case 0X18:
            if (str_R[3]==1)
            {
                        MOTOR_PP=1;
                        TMR0IE = 1;
            }
            else
            {
                        MOTOR_PP=0;
                        TMR0IE = 0;
            }
            break;
//Power do motor passo-a-passo
case 0X19:
```

```
                Power_PP=str_R[3];
                break;

        // Sentido do motor passo-a-passo
        case 0X1A:
                if (str_R[3]==0)
                        SEN_MOTOR_PP=0;
                else
                        SEN_MOTOR_PP=1;
                break;

        // Velocidade do motor passo-a-passo
        case 0X1B:
                velocidade_PP();
                break;

        // tempo de execução PID
        case 0X1C:
                comp=str_R[2];
                        final_val=0;
                        while (comp!=i)
                        {
                                valor=str_R[i+3];
                                final_val=10*final_val+valor;
                                i++;
                        }
                        Tempo_PID=final_val*1000;
                        cria_mensagem(final_val,0x0C);
                break;

        // Valor Kp
        case 0X1D:
                comp=str_R[2];
                        final_val=0;
                        while (comp!=i)
                        {
                                valor=str_R[i+3];
                                final_val=10*final_val+valor;
                                i++;
                        }
                kp=final_val/10000f;
                break;

        // Valor Ki
        case 0X1E:
                comp=str_R[2];
                        final_val=0;
                        while (comp!=i)
                        {
                                valor=str_R[i+3];
                                final_val=10*final_val+valor;
                                i++;
                        }
                ki=final_val/10000f;
                break;

        // Valor Kd
        case 0X1F:
```

```
                        comp=str_R[2];
                                final_val=0;
                                while (comp!=i)
                                {
                                        valor=str_R[i+3];
                                        final_val=10*final_val+valor;
                                        i++;
                                }
                        kd=final_val/10000f;
                        break;
        }
}

//*****************************************************************************
//
//                      FUNCAO QUE CRIA A MENSAGEM A SER ENVIADA
//
//*****************************************************************************
void cria_mensagem(unsigned int val,unsigned char type)
{
        unsigned int str[17];
        unsigned char str_aux[5];
        unsigned char i, j, digito, comprimento;
        unsigned int val_aux=val;

        j = 0;

        do {
                digito = val % 10;
                val /= 10;
                str_aux[j++] = bin2asc(digito);
        } while( val > 0 );

        if (type==0x04)
        {
                digito=RB1;
                str_aux[j++] = bin2asc(digito);
        }

        comprimento=j;

        i=0;
        str[i++]=0b10101010;            //AAh inicio de mensagem
        str[i++]=type;                          //Tipo de mensagem
        str[i++]=comprimento;           //Comprimneto da mensagem

        while( j > 0 ) {
                str[i++] = str_aux[--j];
        }                                                               //mensagem
        str[i++]=0b01010101;            //Fim da mensagem

        envia_mensagem(i,str);

}
```


Título da Dissertação:

Sistema servo-mecanismo didáctico com motores DC e passo-a-passo.

```
//*****************************************************************************
//
//                    FUNCAO DE TESTE DE COMUNICACAO
//
//*****************************************************************************
void comunicacao(void)
{
        unsigned char estado, men;

        estado=1;
        if (str_R[0]!=0xAA)
                estado=0;
        else if(str_R[1]!=0x10)
                estado=0;
        else if(str_R[2]!=0x01)
                estado=0;
        else if(str_R[3]!=0xCC)
                estado=0;
        else if(str_R[4]!=0x55)
                estado=0;
        if(estado==0)
                men=0;
        else
                men=204;

        cria_mensagem(men,0x00);
}


//*****************************************************************************
//
//                    FUNCAO DE LEITURA DA VELOCIDADE DO MOTOR
//
//*****************************************************************************
void velocidade (void)
{
//      unsigned int  freq;

        tempo_cont=0;

        //*************************************************************************************************
        //              Leitura da velocidade do motor
        //*************************************************************************************************
        T1CON = 0x07;
        TMR1H = 0; // clear timer1 high count. stop overflow
        PIR1  &= ~(1<<TMR1IF);  // clear overflow bit.
        TMR1L = 0; // clear timer1 low count

        T2CON=0b01111100;
        TMR2=0;
        PR2=0x0D;
        TMR2IF=0;
        TMR2IE=1;

        while (ok==0);
        freq=(TMR1L+(TMR1H<<8));
        TMR2ON=0;
   TMR2IE=0;
```

```
        ok=0;

        T1CON = 0x00;
//      cria_mensagem(freq,0x04);

}

//*****************************************************************************
//
//                  FUNCAO DE LEITURA DA CORRENTE DO MOTOR
//ADCON0
//bit 5-3 CHS2:CHS0: Analog Channel Select bits
//000 = channel 0, (RA0/AN0)
//001 = channel 1, (RA1/AN1)
//010 = channel 2, (RA2/AN2)
//*****************************************************************************
void corrente (void)
{
        unsigned long int  val_corr;
   ADCON0=0b10001001;       //RA1=1 input

        DelayMs(5);
        ADGO=1;
        while(ADGO);
        ADON=0;
        val_corr=ADRESH<<8;
        val_corr=val_corr|ADRESL;

        cria_mensagem(val_corr,0x05);
}
//*****************************************************************************
//
//                  FUNCAO DE LEITURA DA TENSAO DO MOTOR
//
//*****************************************************************************
void tensao (void)
{
        unsigned long int  val_corr;

        ADCON0=0b10010001;       //RA2=1 input
        DelayMs(5);
        ADGO=1;
        while(ADGO);
        ADON=0;
        val_corr=ADRESH<<8;
        val_corr=val_corr|ADRESL;

        cria_mensagem(val_corr,0x06);
}

//*****************************************************************************
//
//                  FUNCAO DE LEITURA DA VELOCIDADE DO MOTOR PP
// 011 = channel 3, (RA3/AN3)
//*****************************************************************************
void velocidade_PP (void)
{
        unsigned long int  val_corr;
```

```
        ADCON0=0b10001001;      //RA3=1 input
        DelayMs(5);
        ADGO=1;
        while(ADGO);
        ADON=0;
        val_vel=ADRESH<<8;
        val_vel=val_vel|ADRESL;

        cria_mensagem(val_vel,0x0B);
}

//*****************************************************************************
//
//                  FUNCAO DE PID MOTOR DC
//
//*****************************************************************************
void PID (int  vel_pret)
{
        unsigned long  Tempo_aux,Tempo;                         //tempo de amostragem
        float Erro[3];
        Erro[2]=0;              //->Soma dos erros
        Erro[1]=0;              //->Erro anterior
        Erro[0]=0;              // ->Erro actual
        float vel, tem_am;
        int tensao;
        tem_am=tempo*0.05;

        MOTOR_DC=1;                                             // enable motor DC

        Tempo=Tempo_PID/tem_am;
        for(Tempo_aux=0;Tempo_aux<=Tempo;Tempo_aux++)
        {
                velocidade();
                cria_mensagem(freq,0x04);
                vel=freq*120./tem_am;
                Erro[1]=Erro[0];
                Erro[0]=vel_pret-vel;
                Erro[2]=Erro[2]+Erro[0];
                tensao =128+kp*Erro[0]+ kd*(Erro[0]-Erro[1]/tem_am) + ki*Erro[2]*tem_am;
                if (tensao<128)
                        tensao=128;
                if (tensao>255)
                        tensao=255;
                PORTD=tensao;
                cria_mensagem(tensao,0x03);
        }

        MOTOR_DC=0;                                             // disable motor DC
        PORTD=128;

}
```


Título da Dissertação:

Sistema servo-mecanismo didáctico com motores DC e passo-a-passo.

```
//*****************************************************************************
//
//                  FUNCAO PARA CONVERTER 1 DIGITO HEXADECIMAL (4 BITS)
//                  NO CORRESPONDENTE CARACTER ASCII
//
//*****************************************************************************
unsigned char bin2asc(unsigned char num)
{
        num += '0';
        if(num > '9')
                num += 'A'-'9'-1;
        return num;
}

//*****************************************************************************
//
//                  FUNCAO PARA ENVIO DE UMA STRING PARA A PORTA SERIE
//
//*****************************************************************************
void envia_mensagem(unsigned char i, unsigned int *str)
{
        int j=0;
        GIE=0;
        PEIE=0;
        while( i > j ) {
                while( TXIF == 0 );
                TXREG = str[j++];
        }
        GIE=1;
        PEIE=1;
}
```

8.2 Listagem do código do Matlab

```
% Função que abre a comunicação serie
% ComPortName é o nome da porta que vai ser usada para a comunicação
% s1 é o nome atribuido para as comunicacoes entre a PIC e o PC
% s1=-1 não foi possivel abrir a comunicacao com sucesso

function s1=Abrir_porta_serie(ComPortName)

NOK=0;  % Variavel de erro

%Define a porta serie
s1=serial(ComPortName);
s1.BaudRate=19200;

try fopen(s1);  % Tenta abrir a porta serie
catch
   NOK=1;
end
if NOK==1
   fclose(s1);
   msgbox('Nao é possivel abrir a porta','comunicacao porta serie');
   s1=-1;
   return;
end

ret=Test_Comu(s1);  % Chama a funcao que vai testar a comunicacao

if ret==1
   msgbox('Ligacao estabeleciada','comunicacao porta serie');
   return;
end

if ret==-1
   msgbox('Falha na comunicacao','comunicacao porta serie');
   fclose(s1);
```

```
    s1=-1;
    return;
end

% Teste de comunicação
% ret =1 sucesso
% ret =-1 insucesso;
% s1 porta serie

function ret=Test_Comu(s1)
Erro=0;
ret=1;

% Foi definido que a mensaem a enviar seria 11001100 (204 em decimal)
% e é este valor que a PIC está a espera de receber para que haja sucesso
% de mensagem enviada

est=ENVIA_MSG(s1,16,204);   % Funcao para envio de mensagens
if est==-1
    Erro=1;
end

if Erro==0
    pause(0.002);                   % Tempo que demora a enviar a mensagem

    Mensagem=LE_MSG(s1);        %Funcao para leitura de mensagens
    if Mensagem==-1             % Insucesso na leitura de mensagem
        Erro=1;
    else if Mensagem(2)~=0      % Verifica a mensagem enviada pela PIC
        Erro=1;
        else    if Mensagem(4)~=204
        Erro=1;
            end
        end
    end
end

if Erro==1
```

```
    msgbox('Erro no teste de comunicação','comunicacao porta serie');
    ret=-1;
    return;
end



% Função que envia mensagem
% s1 comunicação
% tipo representa o tipo de mensagem
% valor representa o valor a ser enviado
% est se 1 sucesso no envio
%     se -1 erro no envio

function est= ENVIA_MSG(s1,tipo,valor)
est=1;
i=0;

if valor<0
    est=-1;
    msgbox('O conteudo da mensagem tem que ser superior a zero','comunicacao porta serie');
    return;
end

% Mensagem=[inicio tipo comprimento conteudo fim]
Mensagem=[170 1 1 0 85];    %Estrutura geral de uma mensagem

Mensagem(2)=tipo;
if valor<256
% Se o conteudo cabe em 8 bits a estrutura defenida inicialmente mantem-se caso contrario é
% preciso ver o tamanho da mensagem e alterar o tamanho da mensagem
    Mensagem(4)=valor;
else
    while valor>0
        digito=mod(valor,10);
        valor=rounddec(valor/10,0);
        i=i+1;
        array(i)=digito;
    end
```

```
    Mensagem(3)=i;      %comprimento
    for j=0:1:(i-1)
        Mensagem(4+j)=array(i);
        i=i-1;
    end
    Mensagem(5+j)=85;
end



try fwrite(s1,Mensagem);
catch
        est=-1;
end
if est==-1
    msgbox('Nao é possivel enviar a mensagem','comunicacao porta serie');
    s1=-1;
    return;
end



% Função que le mensagens
% s1 porta de comunicacao
% Mensagem se sucesso ntem a estrutura da mensagem recebida
%         se insucesso =-1

function Mensagem= LE_MSG(s1)
mensg=0;
i=1;
sentido=1;  %sentido positivo
pause(0.02);
        while mensg==0
            val=s1.BytesAvailable;
            if val<1
                pause(0.02); %Tempo que demora a enviar a mensagem base
                            val=s1.BytesAvailable;  %Volta a testar se tem valores
                                                    % para serem lidos
                if val<1      % Se não tem dados dá erro e sai da funcao
```

```
        Mensagem=-1;
        msgbox('Nao existem dados a ser lidos','Erro Leitura');
        return;
    end
end

if i==1                 % Mensagem de inicio de mensagem
    val=fread(s1,1);
    if val~=170         % Se valor diferente de inicio dá erro
        Mensagem=-1;
        msgbox('Erro da transmissao da mensagem','Erro Leitura');
        return;
    else
        inicio=170;
    end
end

if i==2                 % Valor referente ao tipo de mensagem
    tipo=fread(s1,1);
end

if i==3                 %Valor do comprimento da mensagem
    comp=fread(s1,1);
end

% Leitura do conteudo da mensagem
if i==4
    if tipo ==4             % Se a mensagem recebida for do tipo velocidade do motor DC
                            % o primeiro byte da mensagem
                            % refere-se ao sentido de rotação
                            % do motor
        sentido=str2double(char(fread(s1,1))');
        if sentido== 0
            sentido=-1;
        end
        comp=comp-1;
    end
    if val<comp
```

```
            pausa=0.0005*comp;      % 1 byte demora cerca de 0.0005 s a ser enviado a uma
                                    % taxa de 19200bit/s
            if pausa <0.01
               pausa=0.01;
            end
            pause(pausa);
            val=s1.BytesAvailable;
            if val<comp
               Mensagem=-1;
               msgbox('Nao existem dados suficientes a serem lidos','Erro Leitura');
               return;
            end
         end
         mens=str2double(char(fread(s1,comp))');
         mens=mens*sentido;
      end

      if i==5        % Leitura do fim de mensagem
         val=fread(s1,1);
         if val~=85
            Mensagem=-1;
            msgbox('Erro da transmissao da mensagem','Erro Leitura');
            return;
         else
            fim=85;
            mensg=1;
         end
      end
      i=i+1;
   end

   Mensagem=[inicio tipo comp mens fim];
```

```
% Funcao que altera o tempo de amostragem
% tem_am-> tempo de amostragem da velocidade em ms
% ret=1 em caso de sucesso
% ret=-1 em caso de insucesso

function [ret,tem_am]=Tempo_Amost(s1,tem_am)
Erro=0;
ret=1;

est= ENVIA_MSG(s1,17,tem_am);
if est==-1
   Erro=1;
end
pause(0.02);
Mensagem= LE_MSG(s1);
if Mensagem==-1
   Erro=1;
else    if Mensagem(2)==1
          tem_am=Mensagem(4);
          return;
       else
          Erro=1;
   end
end

if Erro==1
   ret=-1;
   tem_am=-1;
   return;
end
```

```
% Função que pede e le a valor da corrente
% O valor da corrente vem em mA
% ret =-1 erro na transmissão
% resolucao tem de estar em mA
% s1 porta de comunicacao

function [ret,val_corr] = Recebe_Corrente(s1,resolucao)
Erro=0;
ret=1;

est= ENVIA_MSG(s1,21,0);    %Pede a leitura da corrente
if est==-1
   Erro=1;
end

if Erro==0
   pause(0.02);

   Mensagem=LE_MSG(s1);    %Lê o valor digital
   if Mensagem==-1
      Erro=1;
   else if Mensagem(2)~=5
      Erro=1;
      else
         % valor vai variar entre 0 e 1023 (10 bits de resolução)
         val_corr=(Mensagem(4)*2*resolucao/1023)-resolucao;
         val_corr=rounddec(val_corr,0);  %Elimina as casas decimais
         return;
      end
   end
end

if Erro==1
   ret=-1;
   val_corr=0;
   return;
end
```

```
% Função que pede e le a valor da tensao
% O valor da tensao vem em V
% ret =-1 erro na transmissão
% resolucao tem de estar em V
% s1 porta de comunicacao

function [ret,val_tens] = Recebe_Tensao(s1,resolucao)
Erro=0;
ret=1;

est= ENVIA_MSG(s1,22,0);    %Pede a leitura da tensao
if est==-1
   Erro=1;
end

if Erro==0
   pause(0.015);

   Mensagem=LE_MSG(s1);    %Lê o valor digital
   if Mensagem==-1
      Erro=1;
   else if Mensagem(2)~=6
      Erro=1;
      else
         % valor vai variar entre 0 e 1023 (10 bits de resolução)
         val_tens=(Mensagem(4)*2*resolucao/1023)-resolucao;
         val_tens=rounddec(val_tens,2);
         return;
      end
   end
end

if Erro==1
   ret=-1;
   val_tens=0;
   return;
end
```

```
function Desenha_grafico(amplitude,t,Vel)
hold on
amplitude=amplitude-127;
%legend('Valor Enviado','nº de impulsos /100');

plot(t,amplitude,'b.-');
plot(t,(Vel/10),'r.-');

hold off


% Lê a velocidade motor DC
%devolve a velocidade em rpm

function [ret,vel]=Velocidade_Dc(s1,tem_am,resolucao)
Erro=0;
ret=1;

est= ENVIA_MSG(s1,20,0);
if est==-1
   Erro=1;
end

if Erro==0
   pause(tem_am/1000+0.02+0.05);

   Mensagem=LE_MSG(s1);
   if Mensagem==-1
      Erro=1;
   else if Mensagem(2)~=4
      Erro=1;
      else
         vel=(((1000/tem_am)*Mensagem(4))*60/resolucao);
      end
   end
end

if Erro==1
   ret=-1;
   vel=0;
   return;
end

% Função que gere o ambiente grafico GUI

function varargout = Controlo_Motor_DC(varargin)
% CONTROLO_MOTOR_DC M-file for Controlo_Motor_DC.fig
%      CONTROLO_MOTOR_DC, by itself, creates a new CONTROLO_MOTOR_DC or raises the
existing
%      singleton*.
%
%      H = CONTROLO_MOTOR_DC returns the handle to a new CONTROLO_MOTOR_DC or the
handle to
%      the existing singleton*.
%
%      CONTROLO_MOTOR_DC('CALLBACK',hObject,eventData,handles,...) calls the local
```

```
%      function named CALLBACK in CONTROLO_MOTOR_DC.M with the given input arguments.
%
%      CONTROLO_MOTOR_DC('Property','Value',...) creates a new CONTROLO_MOTOR_DC or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Controlo_Motor_DC_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Controlo_Motor_DC_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Controlo_Motor_DC

% Last Modified by GUIDE v2.5 12-Nov-2009 23:31:47

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Controlo_Motor_DC_OpeningFcn, ...
                   'gui_OutputFcn',  @Controlo_Motor_DC_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Controlo_Motor_DC is made visible.
function Controlo_Motor_DC_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Controlo_Motor_DC (see VARARGIN)

% Choose default command line output for Controlo_Motor_DC
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% This sets up the initial plot - only do when we are invisible
% so window can get raised using Controlo_Motor_DC.
if strcmp(get(hObject,'Visible'),'off')
    plot(rand(1));
end
```

```
%Valores iniciais
set(handles.tempo_a,'String',100);      %Tempo de amostragem
set(handles.amplitude,'String',100);    %Valor da amplitude
set(handles.freq_Sinal,'String',10);    %Valor da frequencia do sinal se aplicavel
set(handles.Run_time,'String',5);       %Valor do tempo de execução
set(handles.Val_vel,'String',0);        %Valor da velocidade
set(handles.Val_ten,'String',0);        %Valor da corrente
set(handles.Val_cor,'String',0);        %Valor da tensao
set(handles.Val_P,'String',0);          %Valor da potencia



global resolucao resolucao_corr resolucao_tens ComPortName dados
ComPortName='COM1';
resolucao=500;                      %Valor do numero de impulsos do encoder
resolucao_corr=2000;                %Valor maximo da corrente em mA
resolucao_tens=15;                  %Valor maximo da tensao em V
clear dados;
% UIWAIT makes Controlo_Motor_DC wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = Controlo_Motor_DC_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global s1 tem_am dados
axes(handles.axes1);
cla;                %Limpa o grafico actual
ENVIA_MSG(s1,18,1);     %Liga o motor DC
i=1;
tem_am= round(str2double(get(handles.tempo_a, 'String'))); %Vai buscar o tempo de amostragen
if tem_am<1
    tem_am=1;
     msgbox('Tempo tem de ser superior a 1','Tempo amostragem');
end
[ret,tem_am]=Tempo_Amost(s1,tem_am);
if ret==-1
    msgbox('Erro na transmissao','Tempo amostragem');
    fclose(s1);
    return;
end
set(handles.tempo_a,'String',tem_am);       %Actualiza o valor da variavel tempo de amostragem

xmax= str2double(get(handles.Run_time, 'String'));
```

```
intervalo=tem_am/1000+0.02+0.05+0.04+0.04+0.04; %soma dos tempos de comunicação
val=s1.BytesAvailable;
if val>0
   fread(s1,val);
end                    %limpar buffer de mensagem recebidas de lixo

% val_amplitude=handles.amplitude
val_amplitude= str2double(get(handles.amplitude, 'String'))+128;
if val_amplitude > 255
   val_amplitude=255;
   set(handles.amplitude,'String',127);
   msgbox('Valor maximo 127','Amplitude');
   return;
else if val_amplitude < 0
       val_amplitude=0;
       set(handles.amplitude,'String',-128);
       msgbox('Valor minimo -128','Amplitude');
       return;
   end
end

val_freq= str2double(get(handles.freq_Sinal, 'String'));

popup_sel_index = get(handles.sinal, 'Value');
i=0;
switch popup_sel_index
   case 1  %IMPULSO
      for t = 0:intervalo:xmax
         if t ==0    %<= val_freq
            tensao=val_amplitude;
         else
            tensao=128;
         end
         i=i+1;
         ret=Actualiza_valores(t,i,tensao,handles);
         if ret ==-1
            return;
         end
       end

   case 2     %DEGRAU
      for t = 0:intervalo:xmax
         tensao=val_amplitude;
         i=i+1;
         ret=Actualiza_valores(t,i,tensao,handles);
         if ret ==-1
            return;
         end
      end

   case 3      %RAMPA
      tensao=128;
      for t = 0:intervalo:xmax
         if val_amplitude>128
            if tensao < val_amplitude
               tensao=tensao+1;
```

```
            else
               tensao=val_amplitude;
            end
         else
            if tensao > val_amplitude
               tensao=tensao-1;
            else
               tensao=val_amplitude;
            end
         end
         i=i+1;
         ret=Actualiza_valores(t,i,tensao,handles);
         if ret ==-1
            return;
         end
      end

   case 4      %Seno
      for t = 0:intervalo:xmax
          tensao=(int16(128+(val_amplitude-128)*sin(2*pi*val_freq*t)));
         i=i+1;
         ret=Actualiza_valores(t,i,tensao,handles);
         if ret ==-1
            return;
         end
      end
end
save dados;
pause(1);
ret=Actualiza_valores(-1,0,128,handles);
if ret ==-1
   return;
end

function ret=Actualiza_valores(t,i,tensao,handles)
global s1 tem_am resolucao resolucao_corr resolucao_tens dados
ret=1;
Erro=0;

est= ENVIA_MSG(s1,19,tensao);
if est ==-1
   Erro=1;
else
   [ret,Vel]=Velocidade_Dc(s1,tem_am,resolucao);
   if ret==-1
      Erro=1;
   else
      if t>= 0
         dados(1,i)=t;
         dados(2,i)=tensao;
         dados(3,i)=Vel;
         Desenha_grafico(tensao,t,Vel);
      end

      set(handles.Val_vel,'String',Vel);

      [ret,val_corr] = Recebe_Corrente(s1,resolucao_corr);
```

```
        if ret==-1
            Erro=1;
        else
            set(handles.Val_cor,'String',val_corr);
            if t>=0
                dados(4,i)=val_corr;
            end
            [ret,val_tens] = Recebe_Tensao(s1,resolucao_tens);
            if ret==-1
                Erro=1;
            else
                if t>=0
                    dados(5,i)=val_tens;
                end
                set(handles.Val_ten,'String',val_tens);
                P=val_tens*val_corr;
                set(handles.Val_P,'String',P);
                return;
            end
        end
    end
end

if Erro==1
    ret=-1;
    fclose(s1);
    return;
end

% --------------------------------------------------------------------
function FileMenu_Callback(hObject, eventdata, handles)
% hObject    handle to FileMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function OpenMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to OpenMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
file = uigetfile('*.fig');
if ~isequal(file, 0)
    open(file);
end

% --------------------------------------------------------------------
function PrintMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to PrintMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
printdlg(handles.figure1)

% --------------------------------------------------------------------
function CloseMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to CloseMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
selection = questdlg(['Close ' get(handles.figure1,'Name') '?'],...
                     ['Close ' get(handles.figure1,'Name') '...'],...
                     'Yes','No','Yes');
if strcmp(selection,'No')
   return;
end
delete(handles.figure1)


% --- Executes on selection change in controlo.
function controlo_Callback(hObject, eventdata, handles)
% hObject    handle to controlo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns controlo contents as cell array
%        contents{get(hObject,'Value')} returns selected item from controlo


% --- Executes during object creation, after setting all properties.
function controlo_CreateFcn(hObject, eventdata, handles)
% hObject    handle to controlo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function tempo_a_Callback(hObject, eventdata, handles)
% hObject    handle to tempo_a (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of tempo_a as text
%        str2double(get(hObject,'String')) returns contents of tempo_a as a double


% --- Executes during object creation, after setting all properties.
function tempo_a_CreateFcn(hObject, eventdata, handles)
% hObject    handle to tempo_a (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on selection change in sinal.
function sinal_Callback(hObject, eventdata, handles)
% hObject    handle to sinal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns sinal contents as cell array
%        contents{get(hObject,'Value')} returns selected item from sinal


% --- Executes during object creation, after setting all properties.
function sinal_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sinal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
set(hObject, 'String', {'Impulso', 'Degrau', 'Rampa', 'Seno'});


function amplitude_Callback(hObject, eventdata, handles)
% hObject    handle to amplitude (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of amplitude as text
%        str2double(get(hObject,'String')) returns contents of amplitude as a double


% --- Executes during object creation, after setting all properties.
function amplitude_CreateFcn(hObject, eventdata, handles)
% hObject    handle to amplitude (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function freq_Sinal_Callback(hObject, eventdata, handles)
% hObject    handle to freq_Sinal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of freq_Sinal as text
%        str2double(get(hObject,'String')) returns contents of freq_Sinal as a double
```

```
% --- Executes during object creation, after setting all properties.
function freq_Sinal_CreateFcn(hObject, eventdata, handles)
% hObject    handle to freq_Sinal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in Port_ser.
function Port_ser_Callback(hObject, eventdata, handles)
% hObject    handle to Port_ser (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns Port_ser contents as cell array
%        contents{get(hObject,'Value')} returns selected item from Port_ser

popup_sel_index = get(handles.Port_ser, 'Value');
global ComPortName
switch popup_sel_index
    case 1
        ComPortName='COM1';
    case 2
        ComPortName='COM2';
    case 3
        ComPortName='COM3';
    case 4
        ComPortName='COM4';
    case 5
        ComPortName='COM5';
    case 6
        ComPortName='COM6';
    case 7
        ComPortName='COM7';
end


% --- Executes during object creation, after setting all properties.
function Port_ser_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Port_ser (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
set(hObject, 'String', {'COM1', 'COM2', 'COM3', 'COM4', 'COM5', 'COM6', 'COM7'});
```

```
function Run_time_Callback(hObject, eventdata, handles)
% hObject    handle to Run_time (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Run_time as text
%        str2double(get(hObject,'String')) returns contents of Run_time as a double


% --- Executes during object creation, after setting all properties.
function Run_time_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Run_time (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global s1
NOK=0;
ENVIA_MSG(s1,19,128);
try fclose(s1);
catch
    NOK=1;
end
if NOK==1
    msgbox('Nao é possivel fechar a porta','comunicacao porta serie');
    s1=-1;
    return;
end
fclose(s1);


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ComPortName s1
s1=Abrir_porta_serie(ComPortName);


% Função de PID implementado na PIC
% Kp Ki Kd parametros do controlador
% tem_am -> Tempo de amostragem da velocidade
```

```
% VEL -> velocidade pretendida
% resolucao= resolução do encoder
% xmax =tempo em segundos de execução do programa

function PID_PIC(kp,ki,kd,VEL,tem_am,resolucao,xmax)
% tem_am=100;
% xmax=10;
% VEL=100;
% resolucao=500;
% kp=0.001;
% ki=0.0001;
% kd=0.01;
intervalo=tem_am/1000;
clear data;
s1=Abrir_porta_serie('COM3');
if s1==-1
   return;
end

[ret,tem_am]=Tempo_Amost(s1,tem_am);

kp=kp*10000;
est=ENVIA_MSG(s1,29,kp);
if est==-1
   fclose(s1);
   return;
end
pause(0.05);
ki=ki*10000;
est=ENVIA_MSG(s1,30,ki);
if est==-1
   return;
end
pause(0.02);
kd=kd*10000;
est=ENVIA_MSG(s1,31,kd);
if est==-1
   return;
end
pause(0.5);
est= ENVIA_MSG(s1,23,VEL);
if est==-1
   return;
end
pause(.05)
i=1;
   pause(intervalo);
for t = 0:tem_am/1000:xmax

   w=0;
   val=s1.BytesAvailable;
   while(val<16)
      pause(0.0001)
      val=s1.BytesAvailable;
      w=w+1;
      if w>100
         break;
```

```
        end
    end
    Mensagem=LE_MSG(s1);
    if Mensagem==-1
        fclose(s1);
        break;
    end
    VEL_AC=(((1000/tem_am)*Mensagem(4))*60/resolucao);

    Mensagem=LE_MSG(s1);
    if Mensagem==-1
        fclose(s1);
        break;
    end
    tensao=Mensagem(4);
    data(1,i)=t;
    data(2,i)=VEL;
    data(3,i)=VEL_AC;
    data(4,i)=tensao;
    i=i+1;
end
i=i-1;

fclose(s1);
hold on
plot(data(1,1:i),data(2,1:i),'b');
plot(data(1,1:i),data(3,1:i),'g');
plot(data(1,1:i),data(4,1:i),'r');
legend('velocidade pretendida','velocidade lida','tensao enviada');
hold off


%Função que implementa o PID atraves do MatLAb
tem_am=100;
intervalo=tem_am/1000+0.02+0.05+0.04;

xmax=10;
VEL=1300;
tensao=128;
kp=20;
kd=20;
ki=20;
Erro(3) = 0; %Soma dos erros
Erro(2) = 0; %Erro anterior
Erro(1) = 0; %Erro actual
s1=Abrir_porta_serie('COM3');
if s1==-1
    return;
end
est= ENVIA_MSG(s1,18,1);
if est==-1
    return;
end
[ret,tem_am]=Tempo_Amost(s1,tem_am);
est= ENVIA_MSG(s1,19,tensao);
if est==-1
    return;
```

```
end
i=1;
for t = 0:intervalo:xmax
    pause(0.02);
    [ret,VEL_AC]=Velocidade_Dc(s1,tem_am,500);
    if ret==-1
        return;
    end
    Erro(2) = Erro(1); %Erro anterior
    Erro(1) = VEL-VEL_AC; %//Erro actual
    Erro(3) = Erro(3)+Erro(1); %Soma dos erros
    tensao = kp*Erro(1)+ kd*(Erro(1)-Erro(2))/tem_am + ki*Erro(3)*tem_am; %PID

    tensao=tensao+128;
    if tensao >255
        tensao=255;
    end
    if tensao <128
        tensao=128;
    end
    est= ENVIA_MSG(s1,19,tensao);
    if est ==-1
          return;
    end
    data(1,i)=t;
    data(2,i)=VEL;
    data(3,i)=VEL_AC;
    data(4,i)=tensao;
    i=i+1;
end
i=i-1;
est= ENVIA_MSG(s1,18,0);
fclose(s1);
hold on
plot(data(1,1:i),data(2,1:i),'b');
plot(data(1,1:i),data(3,1:i),'g');
plot(data(1,1:i),data(4,1:i),'r');
legend('velocidade pretendida','velocidade lida','tensao enviada');
hold off
```

8.3 Esquema da PCB

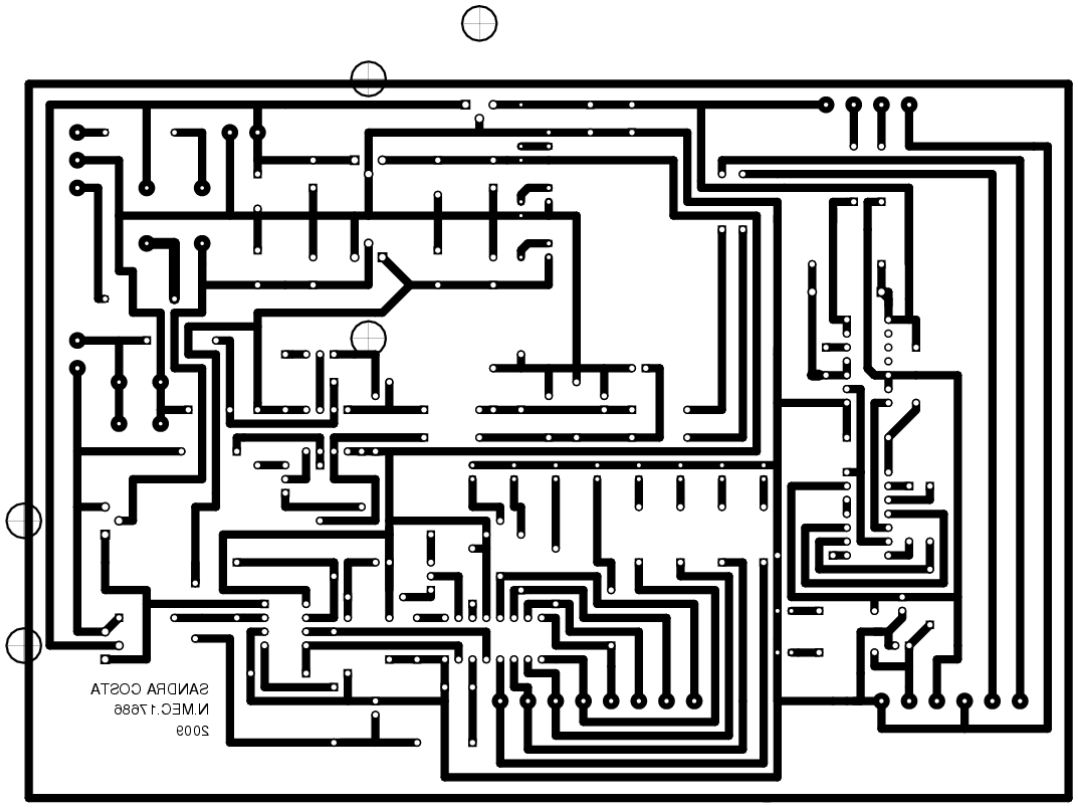


Figura 8.3-1 Motor DC layout

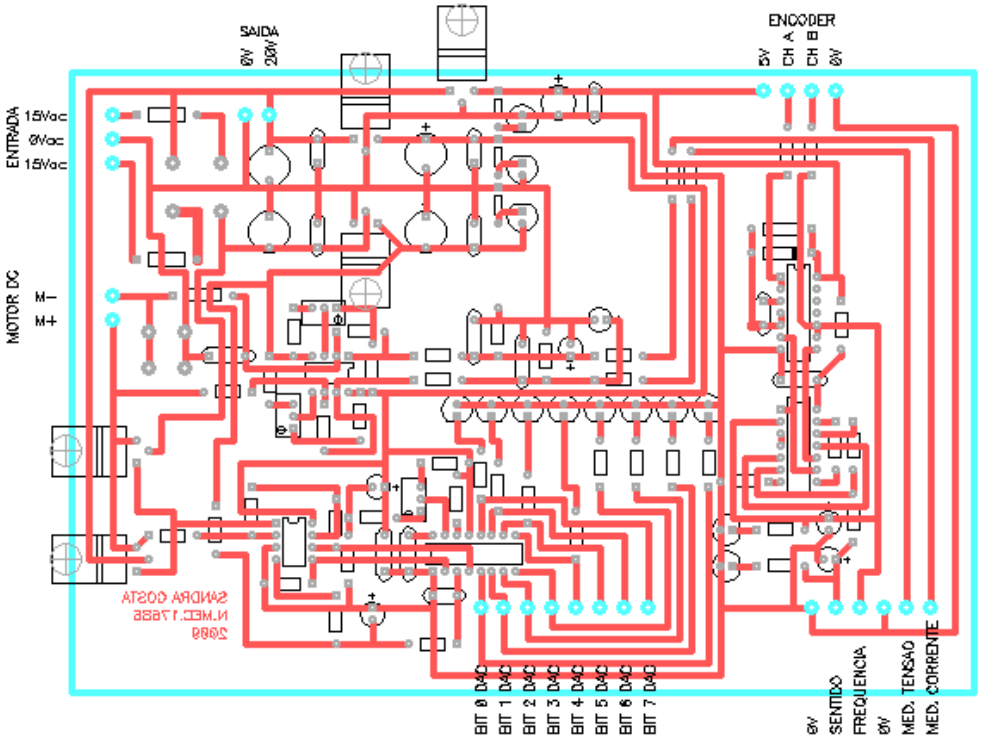


Figura 8.3-2 Motor DC esquema com componentes

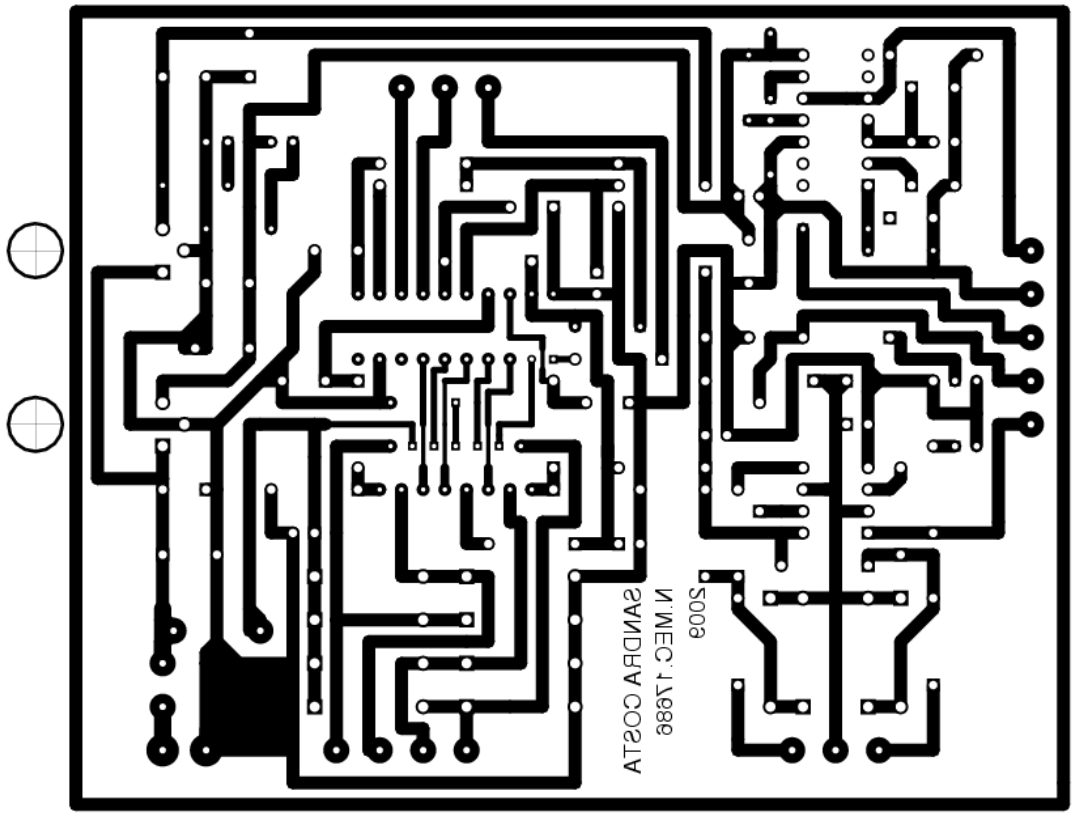


Figura 8.3-3 Motor passo-a-passo layout

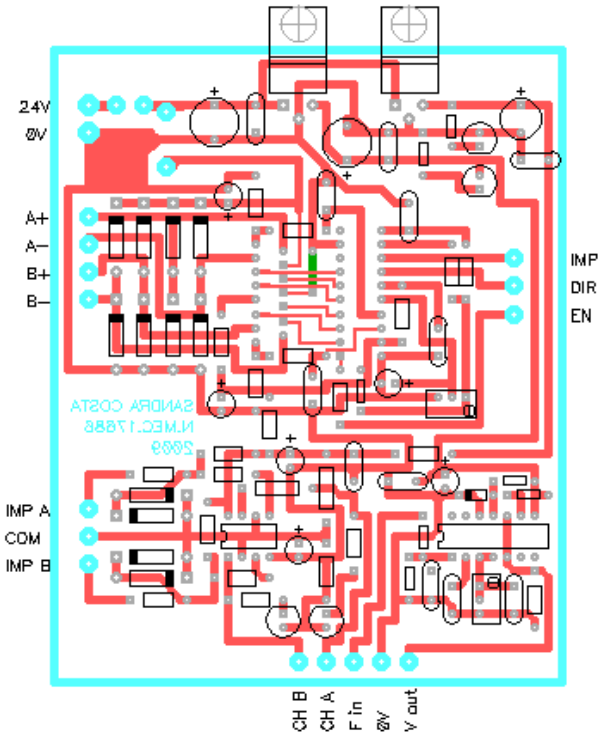


Figura 8.3-4 Motor passo-a-passo esquema com componentes